THÈSE de Doctorat
de l'Université de Versailles - Saint Quentin en Yvelines

Spécialité : Informatique

Présentée par
Nicolas RIVIERRE

en vue de l'obtention du titre de
DOCTEUR DE L'UNIVERSITÉ DE VERSAILLES - ST QUENTIN

Ordonnancement temps réel centralisé, les cas préemptifs et non-préemptifs

Soutenue le 6 Février 1998 devant le Jury :

Président	**J. P. THOMESSE**	**Professeur, CRIN**
Directeur	**G. PUJOLLE**	**Professeur, Université de Versailles**
Rapporteur	**J. CARLIER**	**Professeur, Université de Compiegne**
	K. CHEN	**Professeur, Paris XIII**
Examinateur	**P. MUHLETHALER**	**Ingénieur principal de l'armement**

Remerciements

Mes remerciements vont d'abord à Gérard Le Lann et Laurent Leboucher. Gérard Le Lann pour m'avoir accueilli dans son équipe à l'INRIA. L'intérêt des thèmes d'étude qu'il m'a proposés, son exigence sur les résultats et sa confiance m'ont toujours motivé. Ma gratitude va aussi à Laurent Leboucher du CNET pour sa relecture attentive de mon manuscrit, la valeur de ses conseils et de ses idées sur l'ordonnancement. Tout deux m'ont beaucoup apporté.

Pour les encouragements qu'ils m'ont prodigués, je souhaite ensuite remercier chaleureusement Messieurs Jean Pierre Thomesse, qui a accepté de présider mon Jury, Guy Pujolle, qui a dirigé cette thèse, ainsi que Paul Muhlethater qui m'a initié aux méandres de l'ordonnancement temps réel. Que Messieurs les rapporteurs Jacques Carlier et Ken Chen soient également remerciés pour leurs commentaires avisés qui m'ont grandement aidé à améliorer l'exposé de mes idées.

Le plaisir de travailler sur l'ordonnancement temps réel n'aurait pas été le même sans les fructueuses discussions que j'ai eues avec mes collègues Jean-François Hermant, Laurent George et Marco Spuri. Qu'ils en soient remerciés, comme Jean Bernard Stefani qui m'a offert la possibilité de continuer à explorer ce thème dans son équipe au CNET.

Un dernier mot enfin pour saluer la gentillesse de toutes les personnes que j'ai cotoyées lors de mon passage au projet REFLECS de l'INRIA ainsi que celle de mes proches qui, bien que très réticents à toute évocation de l'ordonnancement temps réel, m'ont supporté lors de ces derniers mois.

Plan

Notations

Audsley	cf. chapitre IV.3.4.3. p. 40.
C_i	Durée d'exécution de τ_i (cf. Définition 1 p. 8).
couple (T,D)	Un couple période-échéance (cf. Définition 27 p. 22 et Définition 36 p. 52).
D_i	Echéance relative de τ_i (cf. Définition 1 p. 8).
$\{D_i\} \gg \{T_i\}$	Lorsque toutes les échéances relatives dans τ sont supérieures aux périodes.
$\{D_i\} \ll \{T_i\}$	Lorsque toutes les échéances relatives dans τ sont inférieures aux périodes.
DM	*Deadline Monotonic* (cf. chapitre IV.3.3.2. p. 37).
EDF	*Earliest Deadline First* (cf. chapitre IV.2.1. p. 28).
ETPN	Ensemble de Trafics Périodiques Non-concrets (cf. Définition 27 p. 22).
$h(t)$	La demande processeur à l'instant *t* (cf. Définition 15 p. 16).
HPF	*Highest Priority First* (cf. Définition 8 p. 13).
L	Taille de la période occupée synchrone du processeur (cf. Définition 17 p. 17).
L_i	Taille maximale de la période occupée de priorité pour τ_i (cf. chapitre III.2.2. p. 19).
$L_i(a)$	Taille d'une *deadline(a+D_i) busy period* (cf. chapitre III.2.2. p. 19).
n	Nombre de tâche d'un trafic τ.
N_{EDF}	Norme *EDF* (cf. Théorème 17 p. 54).
NP-Q	Version non-préemptive non-oisive de l'algorithme *Q* (cf. chapitre II.3.1. p. 12)
N_U	Norme charge (cf. Théorème 17 p. 54).
N'_U	cf. Equation (20) p. 56.
r_i	Pire temps de réponse de τ_i (cf. Définition 5 p. 10).
s_i	L'instant de première activation de ω_i (cf. Définition 1 p. 8).
RM	*Rate Monotonic* (cf. chapitre IV.3.2.2. p. 36).
T_i	Période de τ_i (cf. Définition 1 p. 8).
(T,D)	Un couple période-échéance (cf. Définition 27 p. 22 et Définition 36 p. 52).
TPN	Trafic Périodique Non-concret (cf. Définition 27 p. 22).
U	La charge d'un trafic τ (cf. Définition 9 p. 15).
$W(t)$	La charge cumulée à l'instant *t* (cf. Définition 14 p. 15).
$w_{i,q}$	Taille d'une *level-i busy period* (cf. chapitre III.2.2. p. 19).
$\varepsilon(Q)$	Efficacité de l'algorithme *Q* (cf. Définition 38 p. 53).
δD	Dispersion en échéances : $max(D_i)/min(D_i)$.
δT	Dispersion en périodes : $max(T_i)/min(T_i)$.
Π	Classe d'algorithmes dans le référentiel Σ (cf. Définition 21 p. 20).
Σ	Référentiel d'ordonnancement (cf. Définition 21 p. 20).
τ	Trafic non-concret (cf. Définition 2 p. 9).
τ_i	Tache non-concrète (cf. Définition 1 p. 8).
Υ	Classe de trafics dans le référentiel Σ (cf. Définition 21 p. 20).
ω	Trafic concret (cf. Définition 2 p. 9).
ω_i	Tache concrète (cf. Définition 1 p. 8).

Partie A : Temps réel centralisé : problèmes et outils

I. Introduction

L'ordonnancement temps réel centralisé n'est pas vraiment un problème nouveau. Il a été largement étudié depuis une vingtaine d'années. Les modèles initiaux ont été assouplis et ont conduit à prendre en compte des problèmes additionnels tels que les ressources partagées, les contraintes de précédences entre tâches ou même récemment le contexte réparti. Cette discipline étant assise sur de nombreux résultats en contexte centralisé, on peut se demander si le tour de la question n'a pas été fait.

Motivations

Les critères de choix entre solutions algorithmiques temps réel restent limités. Certains mettent en avant la simplicité d'implémentation des priorités fixes, d'autres la supériorité théorique des priorités dynamiques mais sans vraiment quantifier celle-ci. Notre objectif est de clarifier un peu ce débat en proposant une synthèse des résultats centralisés et quelques éléments pour évaluer les algorithmes.

Le terme “temps réel” couvrira ici la capacité à vérifier, sous certaines hypothèses, des **propriétés** sur le comportement d’un système en présence de contraintes temporelles. L’ordonnancement permet de vérifier de telles propriétés. Précisons que ce terme ne couvrira pas ici les améliorations du système, toujours souhaitables, pour gagner en bande passante et en délais. Citons pour les réseaux informatiques le multiplexage de circuits ou de requêtes ; la transmission multicast ; les profils fonctionnels allégés en termes d’empilements des protocoles...

La propriété de base utilisée en théorie de l’ordonnancement temps réel est la **faisabilité** du problème posé. Aussi insisterons nous sur le sens donné ici à cette faisabilité en présence de contraintes temporelles. L’ordonnancement temps réel s’intéresse aussi typiquement à des propriétés telles que l’**optimalité** des algorithmes ou la **complexité** du calcul des tests de faisabilité. L’optimalité en particulier garantit que tout problème faisable l’est aussi par un algorithme optimal.

La prise en compte de contraintes temporelles applicatives impacte fortement les solutions retenues et leurs propriétés. Bien évidemment, si l’on s ’autorise à trop relaxer ces contraintes, la faisabilité que l’on cherche à établir sort rapidement du cadre de l’ordonnancement temps réel qui nous intéresse ici[1]. Si l’on ne peut les relaxer par contre, la faisabilité peut être démontrée de façon plus ou moins ad-hoc lorsqu’un certain contrôle des ressources est possible. Il faut cependant noter que la recherche de l’optimalité algorithmique trouve souvent peu d’écho lorsque des solutions opérationnelles existent, comme c’est le cas des systèmes embarqués[2] ou à réservation de circuits.

Notre intention ici n’est pas de se focaliser sur cette propriété d’optimalité (dont nous allons d’ailleurs souligner certaines limites), mais de chercher à identifier les solutions algorithmiques les plus performantes, c.à.d. capables de respecter les contraintes temporelles dans un grand nombre de cas faisables. Il faudra alors préciser et prouver dans quelle mesure, et dans quelles situations, un algorithme d’ordonnancement est plus performant (on parlera d’efficacité) qu’un autre. Notons que la dispersion et le manque de points de comparaison offerts par la littérature limite à notre avis cette approche, même sur les modèles les plus simples. C’est donc sur une synthèse des résultats et une évaluation des performances des algorithmes temps réel que nous allons porter notre attention, en précisant et limitant toutefois nos objectifs.

Il est certes possible de montrer l’intérêt d’une solution algorithmique par raffinements successifs du modèle et des contraintes posées. Ceci traduit assez bien l’activité de la communauté scientifique en ordonnancement temps réel et a conduit à de nombreux résultats solides et intéressants, mais aussi à certaines spécialisations[3] qui ne favorisent pas la mise en oeuvre de critères de choix. Citons à nouveau le débat (ou plutôt l’absence de débat) sur les priorités fixes/dynamiques entamé pourtant sur un modèle commun [Liu and Layland 1973].

1. Citons la Qualité de Service (QoS), une exigence récurrente pour les applications multimédia réparties interactives. Actuellement, l’Internet donne un accès immédiat à l’interactivité répartie mais achoppe généralement d’emblée, pour les contraintes temporelles, sur le peu de maîtrise des ressources et de l’algorithmique sous-jacente. Les exigences temporelles font alors place à des controverses autour du *soft real time* (nous n’en connaissons pas de définition qui fasse l’unanimité) ou de façon plus pragmatique à des stratégies de type *best effort*. Notons que les technologies objets du *middleware* (typiquement [ODP 1995], [OMG 1995]...) offrent une voie prometteuse pour envisager une QoS contractuelle (cf. [Leboucher 1998]).

2. Ainsi, la majorité des bus de terrain actuels utilisent la scrutation cyclique, technique largement sous-optimale mais parfaitement opérationnelle pour les applications monolithiques visées. L’examen d’algorithmes plus performants ne peut cependant laisser indifférent si l’on souhaite accéder à un peu de marge de manoeuvre et de flexibilité autrement que par une augmentation de la bande passante (le standard STANAG 3910 utilisé en avionique n’est ainsi qu’une adaptation à 20 Mb/s du bus 1553B, un classique parmi les protocoles à scrutation cyclique).

3. Il suffit de parcourir le sommaire de conférences/journaux sur le temps réel pour s’en convaincre.

Il est aussi possible de montrer l'intérêt d'une solution algorithmique en partant d'un besoin applicatif temps-réel concret. Notre participation[1] à quelques projets de ce type nous a sensibilisé au décalage existant entre les résultats théoriques et leur mise en oeuvre. Encore une fois, il faut de plus très vite spécialiser les modèles et constater que les critères de mesure des performances sont rares, qui permettraient de choisir parmi plusieurs algorithmes.

Ces raisons nous ont conduit[2] à tenter de compléter, en amont, l'information dont nous disposions. Partant d'un problème d'ordonnancement générique, nous avons cherché à identifier, homogénéiser et améliorer quelque peu les solutions pour le résoudre [George et al. 1996], puis à les comparer (coopération CNET/INRIA [Hermant et al. 1996]). Par générique, nous entendons un problème s'énonçant par peu de paramètres, avec toute liberté sur leurs valeurs, et pouvant être spécialisé ultérieurement[3].

Cette approche, complémentaire avec les précédentes, sera celle retenue ici. Nous en résumerons l'idée en disant "synthétisons et comparons avant de spécialiser et non pas l'inverse". L'inconvénient principal porte avant tout sur le modèle que l'on ne pourra aligner directement sur un besoin applicatif. L'intérêt est de proposer un peu de visibilité sur les différentes solutions possibles à un même problème, puis quelques arguments pour les évaluer.

Le problème (cf. chapitre II. p. 7, pour un exposé plus complet)

Nous considérons donc ici le problème posé par le choix d'un **algorithme** d'ordonnancement adapté à un **trafic** temps réel générique. Deux "cas d'école" sont examinés: le cas **centralisé préemptif**, largement étudié (cf. Partie B), et le cas **centralisé non-préemptif**, qui présente des relations partielles avec le précédent (cf. Partie C).

Le trafic temps réel retenu (cf. Définition 2 p. 9) est constitué de tâches concurrentes pour l'accès à un serveur. Chaque tâche (notée τ_i) est réactivable et est caractérisée par trois paramètres : la période T_i, la durée d'exécution C_i et l'échéance relative D_i (cf. Figure 1 p. 9). T_i définit l'interarrivée minimale des activations de τ_i. C_i représente le travail devant être exécuté pour chaque activation de τ_i. Enfin, $t+D_i$ est la date avant laquelle le travail de l'activation de τ_i arrivée en t doit être terminé.

Un tel trafic constitué de n tâches est appelé trafic **non-concret** (noté τ) et peut donner lieu à une infinité de scénarii d'activation, encore appelés trafics concrets (notés ω), en fonction des instants d'activation des tâches.

Un système temps réel peut être caractérisé par des propriétés que nous examinerons ici en combinant les algorithmes préemptifs/non-préemptifs, à priorités fixes/dynamiques. Traditionnellement on considère la **faisabilité** du trafic par un algorithme, l'**optimalité** éventuelle de l'algorithme et la **complexité** du test permettant d'établir la faisabilité. Nous y reviendrons plus formellement dans la suite mais en première approche :

- Faisabilité : établir au préalable[4] la faisabilité d'un trafic τ par un algorithme donné conduit au contrat suivant : *tant que les hypothèses posées seront respectées, les échéances le seront aussi par cet algorithme*. Si les hypothèses sont réalistes, l'intérêt de cette propriété est évident puisque quel que soit le scénario d'activation ω de τ, les échéances seront respectées.

1. Normalisation ETSI/RES10 [Jacquet et al. 1996], [Jacquet et al. 1994] pour des protocoles d'accès multiple sur canal radio ; coopération DRET/DASSAULT/INRIA pour des problèmes transactionnels répartis ; [Hermant et al. 1995] pour des protocoles de type Ethernet déterministe.

2. Dans le cadre du projet REFLECS de l'INRIA.

3. Nous renvoyons à [Cardeira 1994] pour une synthèse des états de l'art existants sur des modèles plus riches que celui utilisé ici, ainsi qu'une proposition de taxonomie pour ces modèles.

4. Nous utilisons le mot préalable, et non pas hors-ligne, car l'analyse de faisabilité peut tout à fait être effectuée durant la vie de l'application. Par exemple lors d'une procédure de contrôle d'admission.

- Optimalité : si un trafic τ n'est pas faisable par un algorithme optimal alors il n'est faisable par aucun autre algorithme. Cette propriété séduisante a cependant plusieurs inconvénients. Les résultats d'optimalité connus sont peu nombreux et peuvent être employés de façon ambiguë lorsque l'on ne précise pas leurs domaines de validité. Cette propriété donne surtout une vision peu exploitable des performances des algorithmes non-optimaux. Nous chercherons donc à l'assouplir.
- Complexité : un test de faisabilité nécessaire et suffisant pseudo-polynomial est généralement jugé acceptable. Des majorants sur les **coûts** des tests seront proposés.

Outils (cf. chapitre III. p. 15, pour un exposé plus complet)

Les échéances étant strictes il faudra nécessairement utiliser une approche pire cas, c.à.d. non probabiliste, pour vérifier la **faisabilité** d'un trafic ainsi que les autres propriétés.

Le concept de **période occupée** caractérise l'activité du processeur (cf. Définition 16 p. 17). En particulier, la période occupée synchrone qui résulte d'une activation simultanée puis périodique des tâches a été très utilisée dans le cas préemptif pour établir la faisabilité des trafics non-concrets τ et borner les intervalles d'étude. La méthode du point fixe permet de calculer la taille (notée L) de cette période.

Dans [Hermant et al. 1996], notre co-auteur L. Leboucher introduit[1] les concepts de référentiel d'ordonnancement et d'efficacité. Les **référentiels d'ordonnancement** (notés Σ, cf. Définition 21 p. 20) en particulier seront très utilisés car ils offrent un formalisme permettant d'énoncer le domaine d'application des résultats d'ordonnancement sans ambiguïtés (c.à.d. l'ensemble des trafics et l'ensemble des algorithmes auxquels se réfère la validité d'une propriété "temps réel"). En jouant sur Σ, il sera ainsi possible de poser la discussion des solutions algorithmiques à un problème sur une base claire et de relativiser l'intérêt d'une propriété telle que l'**optimalité** pour mesurer les performances des algorithmes.

Pour mesurer un peu mieux les performances des algorithmes, nous utiliserons les propriétés de domination et d'efficacité dans Σ. La **domination** relaxe la propriété d'optimalité en énonçant qu'un algorithme en domine un autre si les trafics faisables par le deuxième le sont aussi par le premier (cf. Définition 35 p. 48). L'**efficacité** introduite par L. Leboucher mesure la proximité à l'optimalité de tout algorithme dans Σ (cf. Définition 38 p. 53). Elle s'appuie sur le concept de **demande processeur** qui représente le travail devant nécessairement être effectué dans un intervalle donné (cf. Définition 15 p. 16). Plus précisement, ce concept est traditionnellement utilisé pour établir la faisabilité avec l'algorithme *EDF*, le meilleur théoriquement pour nos problèmes. Dans le cas préemptif, un calcul de norme permet d'étendre ce concept pour établir l'efficacité de tout algorithme comme une mesure de la proximité à l'optimalité d'*EDF*.

Contributions

Le concept de période occupée peut être spécialisé suivant le type de priorité utilisé. La méthode du point fixe permet alors de calculer les instants d'exécution des activations de tâches dans un scénario d'activation ω de τ donné. Très utilisé avec les priorités fixes (on parle alors de *level-i busy period*), nous verrons qu'il est possible de spécialiser ce concept avec les échéances (on parlera de *deadline-d busy period*) utilisées dynamiquement par l'algorithme *EDF*. Nous utiliserons ces notions dans le cas préemptif et non-préemptif pour borner un peu plus les intervalles d'étude, identifier les scénarii d'activation et les inversions de priorités pires cas afin d'en déduire la faisabilité des trafics et calculer les pires temps de réponses des tâches.

1. ces concepts sont développés dans [Leboucher 1998].

Nous insisterons sur la définition de la propriété de faisabilité afin de lever toute ambiguïté sur l'énoncé des autres propriétés qui y font référence. Dans notre cas, un trafic non-concret τ sera dit faisable lorsque tous ces scénarii d'activation ω seront ordonnançables par un même algorithme dans un référentiel d'ordonnancement Σ (cf. Définition 23 p. 21). Les référentiels d'ordonnancement ne se limitent pas cependant à cette définition de la faisabilité. Nous en discuterons brièvement en formalisant d'autres besoins applicatifs temps réel possibles (cf. chapitre III.3.2. p. 22).

Nous avons signalé que l'optimalité est une propriété séduisante mais difficile à interpréter. Un algorithme sera dit optimal s'il peut ordonnancer tout trafic τ faisable dans Σ (cf. Définition 24 p. 21), c.à.d. dans notre cas tel que tous les scénarii d'activation ω de τ soient ordonnançables par un même algorithme dans Σ. Cette définition, qui n'est pas la plus restrictive possible (cf. chapitre III.3.2. p. 22, pour une discussion), nous permettra de rappeler les résultats de l'état de l'art mais aussi d'en mesurer les limites. En particulier, excepté l'algorithme *EDF* dont l'optimalité est très forte pour nos problèmes (cf. Théorème 1 p. 29), nous constaterons que les résultats d'optimalité connus sont vérifiables uniquement dans des référentiels d'ordonnancement très spécialisés, peu généralisables ou comparables. Nous illustrerons notre propos par quelques contre-exemples.

Pour mesurer un peu mieux les performances des algorithmes, quelques relations d'ordre sur la dominance des algorithmes seront identifiées ainsi que quelques impossibilités pratiques. Après avoir rappelé la base théorique de la propriété d'efficacité, nous l'utiliserons pour minorer l'efficacité de *DM*, un algorithme à priorités fixes dominant (cf. chapitre IV.3.3.2. p. 37). Nous tenterons d'interpréter ces résultats, en fonction des trafics soumis et illustrerons notre propos par quelques applications numériques (cf. chapitre X. p. 93). Nous discuterons aussi des limitations rencontrés sur l'optimalité et l'efficacité des algorithmes non-préemptif.

L'utilisation des outils de l'ordonnancement temps réel est donc largement mise en avant. Au-delà, et bien que le modèle étudié reste abstrait, la mise à plat des résultats pour les combinaisons de problèmes préemptifs/non-préemptifs, priorités fixes/dynamiques a pour objectif :

- d'identifier certaines symétries et/ou généralisations possibles afin de proposer quelques extensions aux résultats connus. Citons les *deadline busy periods* avec *EDF* pour limiter les intervalles d'étude ou permettre les calculs de pires temps de réponse en non-préemptifs, l'impact important de l'absence de préemption sur les propriétés d'optimalité et d'efficacité. Cette démarche nous conduira à citer des contributions de nos co-auteurs dans [George et al. 1996] et [Hermant et al. 1996] pour proposer quelques "grilles" de lecture synthétiques des résultats (cf. chapitre XI. p. 103).
- d'attirer l'attention du lecteur sur des problèmes ouverts tels que l'optimalité dans un certain nombre de contextes ou des améliorations possibles sur les calculs.

Notre objectif sera atteint si ce travail peut être considéré comme une introduction à l'algorithmique temps réel et donner quelques éléments de comparaison (visibilité, performances...) sur les solutions algorithmiques à un problème.

Plan de l'étude

Pour éviter un exposé trop linéaire, il nous a semblé utile de structurer notre démarche autour d'un cahier des charges (hypothétique) dont l'objectif est de distinguer :

- le point de vue d'un client qui pose un problème temps réel dont la description fonctionnelle s'accompagne de contraintes temporelles. Ce point de vue ne sera que fictif ici mais pourrait se traduire par une boucle d'interaction quelconque.

- le cahier des charges qui “contractualise” les **hypothèses** permettant de modéliser le problème traité, de fixer le comportement des solutions et les **propriétés** souhaitées.
- le point de vue du fournisseur (qui sera le nôtre) chargé de proposer des solutions et quelques arguments de choix. Loin d’une solution “clé en main”, nous focaliserons sur les **algorithmes d’ordonnancement** et les **outils d’analyse** préalable des propriétés.

Dans la suite de la partie A, nous précisons :

- notre cahier des charges lors du chapitre II. p. 7. Celui-ci couvre les hypothèses (trafics, traitements) et les propriétés examinées (faisabilité des trafics, pires temps de réponse des tâches, optimalité/dominance/efficacité des algorithmes, coût des tests),
- les outils utilisés lors du chapitre III. p. 15. Au-delà des concepts classiques, nous mettrons l’accent sur les périodes occupées, les référentiels d’ordonnancement et les définitions résultantes. En particulier nous discuterons la propriété de faisabilité d’un trafic utilisée ici et à la base des autres propriétés.

La partie B (respectivement C) focalise sur le contexte préemptif (respectivement non-préemptif). Elle procède en trois étapes :

- un état de l’art des résultats connus pour les algorithmes à priorités fixes/dynamiques en termes de faisabilité, de pires temps de réponse, d’optimalité et de dominance. Nous suivrons pour cela l’ordre dans lequel ont été introduits les outils et les résultats en relaxant progressivement les relations particulières entre les paramètres des tâches et en homogénéisant l’énoncé des résultats grâce au concept de référentiel d’ordonnancement,
- des extensions et des discussions sur ces résultats quand cela est nécessaire (et possible). Nous focaliserons plus particulièrement sur des trafics non-concrets τ ne présentant pas de relations particulières entre les paramètres des tâches (on parlera alors de trafics généraux).
- des minorants sur l’efficacité des algorithmes et des majorants sur le coût des tests.

Enfin la partie D propose :

- des applications numériques (cf. chapitre X. p. 93) pour illustrer notre propos,
- une synthèse des résultats (cf. chapitre XI. p. 103) en comparaison des propriétés posées dans notre cahier des charges,

II. Le problème

Après un bref exposé de la démarche choisie, nous spécifions notre cahier des charges (cf. chapitre II.2. p. 8), puis introduisons les solutions qui seront examinées lors des parties suivantes (cf. chapitre II.3. p. 12).

II.1. Cadre méthodologique

Récemment, [Le Lann 1996] analyse les raisons des retards/dépassements de budget/ échecs de projets de dimensions conséquentes[1]. Il identifie des causes communes à ces désagréments et introduit une méthode de **génie système** dont nous allons nous inspirer ici.

D'après cette analyse, il apparaît que le maillon actuellement le plus faible de la chaîne industrielle est le génie système (et non le génie logiciel ou la gestion de projet). Le génie système tel que pratiqué actuellement ne repose pas sur des obligations de preuves, contrairement au génie civil ou au génie electrique en d'autre domaines. Il est pourtant de plus en plus fréquent de se trouver confronté à des besoins applicatifs dont la complexité impose le recours à la preuve. C'est le cas en particulier pour les applicatifs posant directement ou indirectement un ou plusieurs problèmes informatiques de type : Temps Réel (TR), Traitement Distribué (TD) et Tolérant aux Fautes (TF). Leur réunion donne son nom à la **méthode TRDF** proposée dans [Le Lann 1996].

Cette méthode propose non pas de remettre en cause l'ensemble des résultats existants mais de les organiser et de clarifier les responsabilités. Concrètement, avant de passer la main au génie logiciel et à l'implémentation d'une solution, cette méthode propose de débuter un projet informatique par les trois phases suivantes (cf. Table 1 p. 8) :

- la phase de **capture** du besoin applicatif. Il s'agit pour un client et un fournisseur de traduire la description (non, ou partiellement quantifiée) d'un besoin applicatif exprimé par le client en une spécification non ambiguë qui exprime le **cahier des charges** et engage les deux acteurs. Ce cahier des charges définit :
  - les **hypothèses** d'environnement sur lesquelles seront élaborées une, ou plusieurs, solutions (modèle de trafic, de ressource, de topologie, de défaillance, de contraintes...),
  - les **propriétés** requises (qualitatives, quantitatives) pour la(les) solution(s),
- la phase de **conception**. Celle-ci permet au fournisseur de spécifier une solution générique qui résout le problème applicatif posé dans le cahier des charges. Avec la spécification obtenue, on doit notamment fournir la **preuve** que les propriétés attendues seront respectées,
- la phase de **dimensionnement** enfin. Une solution étant établie, il s'agit, à partir d'une quantification des besoins applicatifs fournie par le client, d'établir la faisabilité ou le dimensionnement physique de la solution. Cette phase peut être effectuée à la demande sans remettre en cause la phase de conception, dès lors qu'elle respecte le cahier des charges.

1. ARIANE 5, vol 501 (lanceurs spatiaux européens [Le Lann 1996b]) ; SOCRATE (réservation en-ligne à la SNCF).

Table 1 : méthode TRDF

Phases	Acteurs	*base de départ*	*résultats*
Capture	- client et fournisseur	- besoin applicatif	cahier des charges (hypothèses, propriétés)
Conception	- fournisseur	- cahier des charges - résultats existants	spécification prouvée de la solution
Dimensionnement	- client et fournisseur	- cahier des charges, - spécification de la solution - quantification de l'applicatif	verdict de faisabilité (dimensionnement de la solution)

Comme il n'est pas question de résoudre un problème industriel ici, nous nous inspirerons juste de la démarche TRDF pour donner un fil conducteur à l'exposé de nos résultats. Dans la suite de cette partie, nous capturons un cahier des charges de problèmes temps réel. Dans les parties suivantes, nous dériverons de ce cahier des charges des problèmes d'ordonnancement. Pour chacun la phase de conception examinera les solutions algorithmiques possibles (état de l'art, généralisation, comparaison). La phase de dimensionnement, enfin, sera vue comme une simple illustration de nos résultats par des applications numériques lors de la dernière partie.

II.2. Cahier des charges

Comme il vient d'être dit, le cahier des charges est le résultat de la phase de capture du besoin applicatif. Il comprend un inventaire des **hypothèses** de travail et des **propriétés** attendues de la part d'une solution (cf. Table 2 p. 12 pour un résumé).

II.2.1. Hypothèses

II.2.1.1. Modèle de tâches (les trafics)

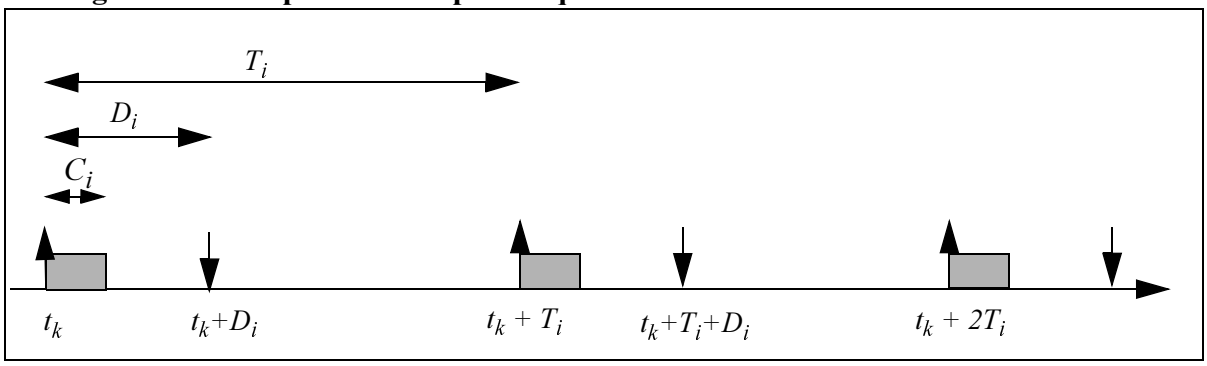
Notre trafic temps réel (noté τ) est constitué de *n* tâches concurrentes pour l'accès à un serveur. τ est qualifié de **non-concret** car il peut donner lieu à une infinité de scénarii d'activation (qualifiés de **concret**s et notés ω) en fixant les instants d'activation des tâches. Plus précisement :

Définition 1 - *Une **tâche non-concrète** τ_i est réactivable et est caractérisée par un triplet (C_i, D_i, T_i). C_i représente le travail devant être exécuté par chaque activation de τ_i. $t+D_i$ est l'échéance absolue avant laquelle le travail d'une activation de τ_i arrivée en t doit être terminé (D_i est donc une échéance relative). Enfin, T_i définit l'interarrivée minimale des activations de τ_i qui donne ainsi lieu à un nombre infini d'**activations**[1] d'un certain travail dont les instants d'exécution seront fixés par un algorithme d'ordonnancement. Notons que :*

- *un scénario d'activation de τ_i (qualifié de tâche **concrète** et noté ω_i) est obtenu par une instanciation particulière des instants d'activation de τ_i. Une tâche non-concrète τ_i peut alors être défini comme l'ensemble de ses scénarii d'activation ω_i possibles.*
- *si pour tout $k \in \mathbb{N}$, la $(k+1)^{ième}$ activation de τ_i arrive à l'instant $t_{k+1} = t_k + T_i$, τ_i est dite **périodique** (cf. Figure 1). Si elle arrive à l'instant $t_{k+1} \geq t_k + T_i$, elle est dite **sporadique** [Mok 1983]. Notons que dans le cas périodique, la simple connaissance de l'intervalle de temps entre l'instant zéro et la première activation de τ_i (noté s_i) permet de déduire un scénario d'activation complet ω_i de τ_i.*

1. On devrait plutôt parler d'occurences. Afin de ne pas surcharger notre terminologie, nous utiliserons exclusivement le terme activation dans la suite, que ce soit pour désigner une occurence de tâche ou son instant d'arrivée.

Figure 1 : Exemple de tâche périodique



Définition 2 - *Un **trafic concret** ω est le scénario d'activation[1] résultant de la superposition des scénarii d'activation de n tâches concrètes. Un **trafic non-concret** τ de n tâches non-concrètes est donc défini comme l'ensemble de ses scénarii d'activation ω possibles. Si de plus les périodes et les échéances relatives sont indépendantes, le trafic non-concret τ est dit **général**[2].*

D'un point de vue plus dynamique, un scénario d'activation ω alimente une file d'attente contenant les tâches déjà activées et en attente d'être exécutées sur un serveur par un algorithme d'ordonnancement. Notons que se limiter à l'analyse d'un ω particulier est contraignant en spécification (il faut connaître les instants d'activation des tâches à l'avance) et surtout réducteur pour les résultats proposés (ils ne sont valables que pour ces instants d'activation). L'inconvénient des trafics non-concrets τ est de devoir considérer à priori l'infinité de scénarii d'activation ω de τ possibles. L'avantage, par contre, est de n'imposer :

- ni hypothèses telles que la connaissance des instants d'activation ou d'une périodicité stricte. Tout au plus, le client s'engage sur une densité maximale d'activation par tâche dont nous verrons que cela peut conduire, parmi la combinatoire de scénarii d'activation, à l'identification de pires cas.
- ni relations particulières entre les paramètres des tâches dans le cas général. Les résultats obtenus en présence de telles relations (par exemple : $\forall i \in [1, n],\ D_i = T_i$) seront vus comme des cas particuliers.

Hypothèse 1 - *Trafics étudiés (à moins que ceci ne soit précisé explicitement) :*

- *trafics non-concrets τ (cf. Définition 2 p. 9).*
- *la surcharge induite par les préemptions, les décisions d'ordonnancement... est incluse dans C_i avec : $\forall i \in [1, n],\ 0 < C_i \leq T_i,\ 0 < C_i \leq D_i$.*
- *les tâches sont indépendantes (pas de précédences) et sans ressources partagées autres qu'un serveur (typiquement un processeur) contrôlé par un algorithme d'ordonnancement.*
- *si $T_i \leq D_i$, une nouvelle demande d'activation de τ_i peut être insérée en file d'attente avant que sa dernière activation ait terminé, ou même commencé, son exécution. Nous considérons dans ce cas que les activations d'une même tâche sont exécutées dans l'ordre d'arrivée.*

1. On utilisera sans distinction les termes ω, "trafic concret" et "scénario d'activation" dans la suite. Notons par ailleur qu'un cas particulier d'un trafic concret est une tâche concrète.
2. Le terme général porte ici sur l'absence de contraintes sur les instants d'activation (τ est non-concret et éventuellement sporadique) et sur les valeurs des paramètres des tâches (les périodes sont indépendantes des échéances). Il ne porte pas sur la prise en compte de contraintes supplémentaires de type ressources ou précédences.

II.2.1.2. Traitements

Le comportement opérationel d'une solution, visible par le client, est le suivant.

Hypothèse 2 - *L'exécution (ou l'ordonnancement) des activations de tâches est :*

- ***préemptive*** *ou* ***non-préemptive*** *(préemptive signifiant que l'activation de tâche en cours d'exécution peut être interrompue par l'arrivée d'une activation plus prioritaire) et toujours* ***non-oisive*** *(l'algorithme d'ordonnancement ne peut être inactif en présence d'activations en file d'attente).*
- *en ligne (c.à.d. sans clairvoyance sur le futur pour prendre les décisions d'ordonnancement). Le scénario d'activation à traiter n'est donc pas connu à l'avance.*
- *non-probabiliste (c.à.d. sans décision basée sur un tirage aléatoire)*
- *bornée par une valeur connue pour établir la faisabilité du trafic. Hypothèse dite* ***synchrone****, par opposition aux modèles partiellement synchrones (les bornes existent mais ne sont pas connues) ou asynchrones (pas de bornes) [Lynch 1996].*

Les hypothèses sur l'environnement sont les suivantes.

Hypothèse 3 - *Environnement :*

- *la topologie est* ***monotrafic, monoserveur*** *par défaut. Typiquement, cette topologie simple permet de modéliser des tâches temps réel s'exécutant sur un processeur (on utilisera indifféremment les mots processeur et serveur dans notre cas centralisé).*
- *le temps est* ***discret*** *c.à.d que les instants d'activation de tâches, les débuts et fins d'exécution se trouvent sur des tics d'horloges (la granularité temporelle). Si les paramètres décrivant les tâches sont eux aussi exprimés par des multiples du tic d'horloge, [Baruah et al. 1990] montrent alors qu'il n'y a pas de perte de généralité à se restreindre à des ordonnancements discrets.*
- *les traitements sont* ***fiables*** *(pas de défaillance du serveur...).*

II.2.2. Propriétés

Enonçons maintenant les propriétés attendues par le client d'une solution à son problème. Ces propriétés sont pour l'instant des propositions, à démontrer lors de la phase de conception dans un environnement donné.

II.2.2.1. Faisabilité

Définition 3 - *L'ordonnancement d'un trafic concret* ω *est dit* ***valide*** *si et seulement si aucune activation de tâche ne rate son échéance absolue.*

Définition 4 - *Un trafic concret* ω *est dit* ***faisable*** *si et seulement si au moins un ordonnancement valide peut être obtenu par un algorithme d'ordonnancement. Similairement, un trafic non-concret* τ *est dit faisable si et seulement si toute instanciation concrète* ω *de* τ *est faisable.*

Définition 5 - *Soit* τ *un trafic non-concret.* $\forall i \in [1, n]$, r_i *est le* ***pire temps de réponse*** *de la tâche* τ_i *parmi les scénarii d'activation* ω *possibles de* τ*, c.à.d. la plus grande durée prise par une quelconque de ces activations entre son instant d'arrivée et son instant de fin d'exécution. La valeur de* r_i *dépend donc du trafic et de l'algorithme d'ordonnancement utilisé.*

Nous attirons l'attention du lecteur sur le fait que la définition précédente et usuelle de la faisabilité doit être maniée avec précaution. En particulier pour les trafics non-concrets τ que nous étudierons (cf. Hypothèse 1 p. 9). Nous l'adoptons en première approche mais en préciserons l'énoncé grâce au concept de référentiel d'ordonnancement (cf. chapitre III.3. p. 20). D'autres énoncés possibles de la faisabilité, et leurs implications, seront discutés lors du chapitre III.3.2. p. 22. La propriété de la faisabilité s'énonce généralement sous deux formes :

Propriété 1 - *Le trafic τ est faisable.*

Propriété 2 - *Les pires temps de réponse des tâches du trafic τ sont inférieurs aux échéances relatives respectives.*

L'analyse de faisabilité doit donc être vue comme une boite noire (un test) capable de rendre un verdict : oui ou non, le problème ainsi quantifié est-il faisable et/ou quels sont les pires temps de réponse? En cas de réponse positive il existe donc un algorithme d'ordonnancement qui respecte les échéances quel que soit ω de τ. Notons que ces deux formulations sont équivalentes en termes de faisabilité mais que la Propriété 2 est plus informative. Elle est aussi plus coûteuse à mettre en oeuvre et donc pas toujours pertinente, sauf s'il n'y pas d'autre approche possible.

A titre d'exemple elle est toujours utilisée avec les priorités fixes et est généralisée dans l'analyse holistique dont l'objectif est d'établir la faisabilité de bout en bout d'un système réparti de type chaîne de traitement [Tindell 1995]. La technique consiste à calculer les pires temps de réponses engendrés par chaque maillon de la chaîne et à utiliser cette information (sous forme de "gigue") lors du maillon suivant.

II.2.2.2. Qualification

Il s'agit ici de qualifier les algorithmes d'ordonnancement. Nous considérerons les critères d'efficacité des algorithmes (extension de l'optimalité) et de coût des tests associés.

Définition 6 - *Un algorithme d'ordonnancement est dit **optimal** si et seulement si celui-ci génère un ordonnancement valide pour tout trafic faisable.*

Cette définition de l'optimalité est liée à la faisabilité (cf. Définition 4 p. 10), elle est donc ambiguë elle aussi et nous y reviendrons lors du chapitre III.3. p. 20. Notons que si plusieurs quantifications du problème sont envisagées, mais non spécifiées à l'avance, choisir l'algorithme optimal peut être intéressant pour assurer une certaine pérénité aux solutions.

Notre opinion est cependant que ce critère donne une vision trop absolue de la qualité d'un algorithme *P*. En effet, celui-ci est optimal ou ne l'est pas. Est-il mauvais pour autant s'il ordonnance correctement bon nombre des quantifications possibles ? Afin d'assouplir cette vision, L. Leboucher introduit dans [Hermant et al. 1996] le critère d'efficacité de *P*, qui sera formalisé lors du chapitre VI.1.2. p. 52. Son objectif est de déterminer la meilleure borne sous laquelle tous les trafics seront faisables par *P*. En première approche :

Définition 7 - $\varepsilon(P)$*, l'**efficacité** d'un algorithme d'ordonnancement P est une mesure de la proximité à l'optimalité, où* $0 \leq \varepsilon(P) \leq 1$ *et (*$\varepsilon(P) = 1$*) => P est optimal*[1]*.*

D'où l'on dérive la propriété suivante sur l'efficacité/optimalité de *P* :

Propriété 3 - *Si P n'est pas optimal,* $\varepsilon(P)$ *doit être connue ou minorée.*

1. Nous verrons que l'inverse sur cette implication n'est pas nécéssairement vérifiée en fonction du problème posé.

Notons que l'efficacité maximale est généralement atteinte par des algorithmes d'ordonnancement à priorités dynamiques. Pour des raisons diverses les produits actuels font généralement appel à des priorités fixes, moins efficaces, quand ils ne se limitent pas à un multiplexage temporel statique. Nous examinerons l'efficacité de ces algorithmes afin de savoir dans quelles conditions ils ne s'éloignent pas trop de l'optimal.

En complément, les conditions de faisabilité (encore qualifiées de tests) étant dépendantes de l'algorithme choisi, il est utile de considérer leurs coûts afin de qualifier une solution. Pour des raisons pratiques la littérature focalise sur des tests pseudo-polynomiaux (c.à.d. d'un coût borné par un polynôme fonction du codage des variables du problème et de bornes sur ces variables). Nous ne ferons pas appel à des notions de complexité plus sophistiquées ici.

Propriété 4 - *Les tests sont pseudo-polynomiaux.*

II.2.2.3. Résumé

Table 2 : Cahier des charges

Hypothèses	Trafics	Trafics non-concrets τ (cf. Hypothèse 1 p. 9)
	Exécution	Préemptif ou non-préemptifs, non-oisif, en-ligne et non-probabiliste (cf. Hypothèse 2 p. 10)
	Environnement	Monotrafic monoressource, temps discret, traitements fiables (cf. Hypothèse 3 p. 10)
Propriétés	Pour un trafic particulier	Faisabilité et pires temps de réponse (cf. Propriété 1 p. 11 et Propriété 2 p. 11)
	Solution algorithmique	Efficacité/optimalité (cf. Propriété 3 p. 11) tests pseudo-polynomiaux (cf. Propriété 4 p. 12)

Nous ne discuterons pas de critères techniques (implémentablité, vitesses de changement de contextes, ...) ou commerciaux ("*lobbying*", prix, ...). La vogue est aux priorités fixes mais la gestion de priorités dynamiques ne sera sans doute pas toujours considérée comme une difficulté, cf. [Yuan 1991]. Par contre, nous allons voir que la meilleure efficacité théorique des algorithmes à priorités dynamiques est à relativiser en fonction des trafics proposés (cf. chapitre VI.1. p. 51).

II.3. Phase de conception

Cette phase conduit à spécifier et à prouver une (ou des) solution(s) pour tout problème posé dans le cahier des charges. Nous introduisons ici brièvement les algorithmes d'ordonnancement et les outils d'analyse de propriétés qui seront examinés dans la suite.

II.3.1. Algorithmes d'ordonnancement

Par algorithme d'ordonnancement temps réel, au delà du comportement visible par le client (cf. Hypothèse 2 p. 10), on entend ici la technique d'assignation des priorités et le *dispatcheur*.

Définition 8 - *Algorithmes d'ordonnancement :*

- *les technique d'assignation des priorités sont nombreuses, par exemple :*
  - *cycliques (round robin, TDMA, jeton et autres multiplexages temporels). Ces techniques qui figent un cycle d'exécution ne seront pas détaillées mais couvertes par les preuves d'optimalité proposées. Elles sont largement sous optimales face à nos trafics non-concrets.*
  - *à priorités fixes. Une priorité est attribuée hors-ligne par tâche, le problème étant de déterminer la meilleure assignation de priorité possible parmi toutes les permutations possibles. Toutes les activations d'une même tâche ayant la même priorité, elles seront ordonnancées par ordre d'apparition.*
  - *à priorités dynamiques. Chaque activation de tâche a une priorité propre assignée en-ligne selon un critère (on se doute qu'en temps réel l'ordre FIFO sera dominé par celui des échéances absolues). Toutes les activations de tâches ayant la même priorité seront ordonnancées par ordre d'apparition.*
- *le dispatcheur est en-ligne, non-oisif et HPF (Highest Priority First), c.à.d. qu'il exécute sans attente les activations de tâches présentes en file d'attente par ordre de priorité, s'il n'y a pas de blocage (absence de préemption, verrou...).*

Cette définition usuelle d'un algorithme d'ordonnancement peut conduire à des inversions de priorités si un blocage résulte, par exemple, de l'absence de préemption ou d'un mécanisme de contrôle d'accès à une ressource partagée. L'avantage est alors de pouvoir cerner la déviation de l'ordonnancement *HPF* pur (préemptif sans blocages) en identifiant les pires inversions de priorités possibles.

L'inconvénient relatif est de décrire le problème en faisant porter par la solution (le *dispatcheur*) l'expression de contraintes qui semblent plus naturellement faire partie des hypothèses du client (partage de ressource, accès décentralisé, comportement préemptif/non-préemptif...).

La convention utilisée pour nommer les algorithmes d'ordonnancement s'inspire de la littérature. Dans le cas :

- **préemptif, non-oisif**, nous utiliserons simplement la technique d'assignation de priorités pour nommer l'algorithme d'ordonnancement (les autres termes sont des options par défaut). Par exemple :
  - *HPF-RM-Préemptif-NonOisif* sera noté *RM* (cf. chapitre IV.3.2.2. p. 36).
  - *EDF-Préemptif-NonOisif* sera noté *EDF* (cf. chapitre IV.2.1. p. 28).
- **non-préemptif et non-oisif**, nous ajouterons le préfixe *NP*. Par exemple *EDF-NonPréemptif-NonOisif* sera noté *NP-EDF* (cf. chapitre VII.2. p. 69).

II.3.2. Analyse de propriété

L'analyse préalable de la **faisabilité** d'un trafic conduira le fournisseur à établir un test permettant de vérifier la Propriété 1 p. 11 ou la Propriété 2 p. 11. Typiquement, ce test sera basé sur l'identification des scénarii d'activation pires cas et de bornes sur l'intervalle d'étude. Suivant le cas le test sera une condition de faisabilité :

- nécessaire et suffisante si : Propriété <=> condition.
- ou à moindre coût, suffisante si : Propriété <= condition.
- ou parfois (lors d'étapes intermédiaires), nécessaire si : Propriété => condition.

L'analyse préalable de l'**optimalite/efficacité** d'un algorithme d'ordonnancement (cf. Propriété 3 p. 11) conduira typiquement le fournisseur à utiliser des raisonnements de permutation pour établir l'optimalité et à des calculs de bornes pour mesurer l'efficacité.

L'analyse préalable du **coût** de mise en oeuvre des tests (cf. Propriété 4 p. 12) conduira le fournisseur à prouver que ceux-ci sont pseudo-polynomiaux en majorant les coûts de mise en oeuvre et en utilisant des bornes sur les variables du problème.

III. Outils

Nous nous constituons ici une "boite à outils" pour traiter nos problèmes d'ordonnancement. Ces outils vont nous permettre de préciser les propriétés "temps réel" examinées et sont pour la plupart issus de la littérature ou des deux rapports co-signés [George et al. 1996] et [Hermant et al. 1996]. Plus précisement, nous introduisons :

- quelques concepts de base en ordonnancement temps réel (cf. chapitre III.1. p. 15).
- les périodes occupées qui seront largement utilisées (cf. chapitre III.2. p. 16).
- les référentiels d'ordonnancement et les définitions résultantes (cf. chapitre III.3. p. 20).

Quelques concepts dépendants du contexte ou faisant appel à des démonstrations spécifiques seront introduits ponctuellement dans la suite.

III.1. Concepts de base

Un trafic non-concret τ (cf. Définition 2 p. 9) est caractérisé par les définitions suivantes :

Définition 9 - *[Liu and Layland 1973].* $\boldsymbol{U} = \sum_{i=1}^{n} C_i/T_i$ *est la* ***charge*** *maximale que* τ *peut imposer à un serveur.*

Notons que dans le cas centralisé, une condition nécessaire évidente pour la faisabilité de τ par tout algorithme d'ordonnancement est que $U \leq 1$.

Définition 10 - *Le* ***scénario critique*** *d'une tâche est le scénario d'activation* ω *de* τ *conduisant au pire temps de réponse de cette tâche.*

Notons que ce scénario critique dépend de l'algorithme d'ordonnancement utilisé.

Définition 11 - *Un* ***instant synchrone*** *(ou* ***scénario synchrone****) est un instant où toutes les tâches de* τ *sont activées simultanément puis périodiquement ensuite.*

A titre d'exemple, pour un instant synchrone 0 nous avons $\forall i \in [1, n]$, $s_i = 0$.

Définition 12 - *[Liu and Layland 1973].* τ *utilise pleinement un serveur pour un algorithme d'ordonnancement P s'il est faisable par P et si l'augmentation de la durée d'exécution d'une de ces tâche conduit* τ *à n'être plus faisable par P. Nous dirons que ce trafic est* ***P-critique(τ)****.*

Définition 13 - *[Leung and Merril 1980]. Pour tout* τ, $\boldsymbol{P} = ppcm_{1 \leq i \leq n}\{T_i\}$ *est la* ***période de base****, c.à.d. un cycle tel que le scénario d'activation des tâches se reproduit similairement si les activations des tâches sont périodiques.*

Notons que même en présence d'un nombre limité de tâches la valeur de P peut être grande si les périodes sont premières entre elles.

Définition 14 - *[Baruah et al. 1990b]. Soit le scénario synchrone en 0 sur* τ*, la* ***charge cumulée*** $\boldsymbol{W(t)}$ *(respectivement* $\overline{\boldsymbol{W}}(\boldsymbol{t})$ *) représente en tout* $t \geq 0$ *la somme des durées d'exécution de toutes les activations de tâches arrivéedans l'intervalle* $[0, t[$ *(respectivement* $[0, t]$*). Nous avons :*

$$W(t) = \sum_{i=1}^{n} \lceil t/T_i \rceil C_i$$

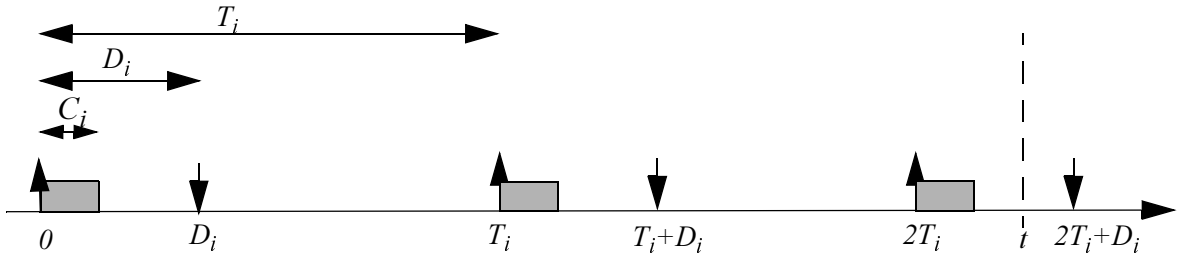
$$\overline{W}(t) = \sum_{i=1}^{n} (1 + \lfloor t/T_i \rfloor) C_i \ .$$

Définition 15 - *[Baruah et al. 1990b]. Soit le scénario synchrone en 0 sur* τ*, la* ***demande processeur*** $\boldsymbol{h(t)}$ *représente, en tout* $t \geq 0$*, la somme des durées d'exécution de toutes les activations de tâches arrivées dans l'intervalle* $[0, t]$ *et dont les échéances absolues sont dans cet intervalle. Nous avons :*

$$h(t) = \sum_{i=1}^{n} max(0, 1 + \lfloor (t - D_i)/T_i \rfloor) C_i = \sum_{D_i \leq t} (1 + \lfloor (t - D_i)/T_i \rfloor) C_i .$$

La Figure 2 illustre, pour une tâche d'indice *i*, ces deux derniers concepts largement utilisés dans la suite. Comme nous pouvons le voir, la charge cumulée ne fait appel qu'aux instants d'activations des tâches alors que la demande processeur fait appel aux instants d'activations et aux échéances absolues des tâches. Dans les deux cas, pour tout $t \geq 0$, ces fonctions sont discontinues, positives et croissantes.

Figure 2 : Charge cumulée et demande processeur en *t*, d'une tâche d'indice *i*



Pour cet exemple, $D_i < T_i$ et $2T_i < t < 2T_i + D_i$, nous avons donc:

$$W_i(t) = \lceil t/T_i \rceil C_i = 3C_i$$

$$h_i(t) = max(0, 1 + \lfloor (t - D_i)/T_i \rfloor) C_i = 2C_i$$

Dans le cas centralisé, quel que soit l'algorithme d'ordonnancement *P*, une condition de faisabilité **nécessaire** pour un trafic non-concret τ s'énonce trivialement comme suit. Nous verrons que cette condition est aussi suffisante pour l'algorithme *EDF* (cf. chapitre V.1. p. 41) et qu'il est possible d'en déduire une mesure de l'efficacité des algorithmes (cf. chapitre VI.1.2.2. p. 54).

Lemme 1 - *Soit* τ*, un trafic non-concret. Si* τ *est faisable alors:*

$$\forall t, h(t) \leq t .$$

Preuve. Construisons ω, un scénario d'activation possible de τ. Soit *0*, un instant synchrone pour toute tâche τ_i de τ. Comme l'inter-arrivée minimale de τ_i est T_i, il peut donc y avoir $max(0, 1 + \lfloor (t - D_i)/T_i \rfloor)$ activations de τ_i arrivées, et ayant une échéance absolue, dans l'intervalle $[0, t]$. Ainsi quelque soit t positif, il doit être possible d'exécuter $\sum_{i=1}^{n} max(0, 1 + \lfloor (t - D_i)/T_i \rfloor) C_i$ dans $[0, t]$ si par hypothèse τ est faisable par *P*. ❐

III.2. Période occupée

Nous introduisons ici ce concept, d'abord pour caractériser l'activité du serveur (typiquement un processeur) puis en le spécialisant ensuite en fonction des priorités utilisées (cf. chapitre III.2.2. p. 19).

III.2.1. Période occupée du processeur

Définition 16 - *Une **période occupée du processeur** est un intervalle de temps durant lequel le processeur est continuellement occupé à exécuter des activations de tâches.*

En présence d'un scénario d'activation ω, un algorithme d'ordonnancement non-oisif donne lieu une succession de périodes occupées et de périodes inoccupées du processeur, c.à.d. toute période telle qu'aucune activation de tâche ne soit en attente d'être servie[1].

Parmi toutes les périodes occupées possibles, nous allons plus particulièrement nous intéresser à la période occupée résultant du scénario synchrone (cf. Definition 11 p. 15). Ce concept, implicitement introduit dans [Liu and Layland 1973], a été largement utilisé par la suite, aussi bien pour l'étude des algorithmes à priorités fixes qu'à priorités dynamiques.

Définition 17 - *[Liu and Layland 1973]. Soit* τ, *un trafic non-concret quelconque et le scénario d'activation synchrone en 0 des tâches de* τ. *La période occupée du processeur allant de 0 jusqu'à l'instant* L, *marquant le début de la prochaine période inoccupée du processeur, est appelé la **période occupée synchrone du processeur**.*

Cette période occupée synchrone du processeur est importante car elle est, comme nous allons le voir, la plus contraignante en terme de faisabilité dans le cas préemptif. La valeur de L est donné par la charge cumulée $W(L)$ (cf. Definition 14 p. 15). Elle se calcule récursivement comme suit : étant donné un intervalle $[0, t[$ avec un instant synchrone en 0 et une activation périodique ensuite, l'idée est de comparer la charge cumulée $W(t)$ avec la longueur t de l'intervalle. Si $W(t)$ est plus grand que t alors, d'après la Définition 17 p. 17, la durée de la période occupée synchrone du processeur est au moins égale à $W(t)$. L'argument est reconduit sur $W(t)$, $W(W(t))$, … , jusqu'à ce qu'on trouve une valeur égale à la précédente (c.à.d. un passage en file vide, éventuellement ponctuel car l'intervalle est ouvert à droite). Formellement, L est le point fixe de l'équation récursive suivante :

$$\begin{cases} L^{(1)} = \sum_i C_i \\ L^{(m+1)} = W(L^{(m)}) = \sum_{i=1}^{n} \left\lceil L^{(m)} / T_i \right\rceil C_i \end{cases} \quad (1)$$

On le voit, le calcul est arrêté lorsque deux valeurs consécutives $L^{(m)}$ et $L^{(m+1)}$ sont égales. L est alors égale à $L^{(m)}$ et nous trouvons bien la première solution de :

$$L = W(L). \quad (2)$$

[Liu and Layland 1973], [Katcher et al. 1993] (respectivement [Ripoll et al. 1996]) montrent que la période occupée synchrone du processeur est la plus longue période occupée possible pour tout trafic non-concret τ tel que $\forall i \in [1, n]$, $T_i = D_i$ (respectivement $\forall i \in [1, n]$, $T_i \leq D_i$). Nous étendons facilement ce résultat en présence de trafics généraux (c.à.d. non-concrets sans relations sur les paramètres T_i et D_i des tâches).

Lemme 2 - *[George et al. 1996]. Soit* τ, *un trafic non-concret quelconque. Si* L *est la longueur de la période occupée synchrone du processeur et* L' *la longueur de toute autre période occupée du processeur, alors* $L \geq L'$.

Preuve. Soit une période occupée du processeur démarrant à l'instant t et de longueur L'. Par définition, cette période est nécessairement précédée par une période inoccupée du processeur, l'instant t coïncide donc avec une activation de tâche et toutes les autres tâches ont une première activation à un instant supérieur ou égal à t.

1. Notons qu'une période inoccupée peut avoir une durée nulle si une activation de tâche arrive en fin d'une période occupée.

Modifions maintenant ce scénario d'activation par un décalage vers la gauche des activations de tâches, de façon à obtenir le scénario d'activation synchrone en *t* (cf. Definition 17 p. 17). Comme $W(L')$, la charge cumulée entre *t* et $t + L'$ ne peut pas avoir diminué suite à ce décalage, alors la longueur de la période occupée du processeur ainsi obtenue ne peut pas avoir diminuée. En d'autres termes $L \geq L'$. ❒

Notons que la propriété établie par ce lemme ne dépend pas de l'algorithme d'ordonnancement choisi, préemptif/non-préemptif, à priorités fixes/dynamiques, dès lors que celui-ci est non-oisif. En fait, la durée d'une période occupée du processeur dépend uniquement du scénario d'activation des tâches.

Voyons maintenant que l'éq. (1) est convergente.

Lemme 3 - *[Hermant et al. 1996]. Si la charge* $U = \sum_i C_i/T_i \leq 1$*, alors l'éq. (1) converge en un nombre fini d'itérations.*

Preuve. Soit *P* la période de base multiple de toutes les périodes (cf. Définition 13 p. 15). Nous avons :

$$W(P) = \sum_i \lceil P/T_i \rceil C_i = P\sum_i C_i/T_i = U \cdot P \text{ (avec } U \cdot P \leq P \text{ car } U \leq 1).$$

Si par hypothèse, $L^{(m)} \leq U \cdot P$, alors :

$$L^{(m+1)} = \sum_i \lceil L^{(m)}/T_i \rceil C_i \leq \sum_i \lceil (U \cdot P)/T_i \rceil C_i \leq \sum_i \lceil P/T_i \rceil C_i = U \cdot P.$$

Comme $L^{(1)} = \sum_i C_i \leq \quad W(P) = U \cdot P$ il vient par récurrence que $L^{(m)}$ est bornée par $U \cdot P$

Si par hypothèse, $L^{(m+1)} \geq L^{(m)}$, alors :

$$L^{(m+2)} = \sum_i \lceil L^{(m+1)}/T_i \rceil C_i \geq \sum_i \lceil L^{(m)}/T_i \rceil C_i = L^{(m+1)}.$$

Comme $L^{(2)} = \sum_i \lceil L^{(1)}/T_i \rceil C_i \geq \sum_i C_i = L^{(1)}$, il vient par récurrence que $L^{(m)}$ est croissante.

$L^{(m)}$ est bornée et croissante, elle est donc convergente. De plus, à chaque itération intermédiaire, la valeur de $L^{(m)}$ augmente d'au moins $min_i(C_i)$. La valeur finale de L, atteinte lorsque $L^{(m+1)}$ est égale à $L^{(m)}$, est donc atteinte en un nombre fini d'itérations. ❒

Si par hypothèse, la charge $U \leq c < 1$, alors la complexité du calcul de L est pseudo-polynomiale. [Leboucher and Stefani 1995] et [Spuri 1996] établissent ce résultat à l'aide de quelques manipulations algébriques. Dans [Hermant et al. 1996], ce calcul est précisé.

Lemme 4 - *[Hermant et al. 1996]. Si l'on pose les constantes* $\delta T = max(T_i)/min(T_i)$, $U \leq c < 1$ *et* ξ *le coût d'une opération élémentaire, la complexité du calcul de L est pseudo-polynomiale en* $O(4n^2\lceil (c/(1-c)) \cdot \delta T \rceil \xi)$.

Preuve. Le coût pour résoudre l'Equation (1) p. 17 est égal au nombre d'itérations que multiplie le coût de chaque itération.

Un majorant possible sur le nombre d'itérations est égal à $\sum_i \lceil L/T_i \rceil$, c.à.d. au nombre d'activations de tâches dans l'intervalle $[0, L[$. Or:

$$\left(L = \sum_i \left\lceil \frac{L}{T_i} \right\rceil C_i \leq \sum_i \left(1 + \frac{L}{T_i}\right) C_i = \sum_i C_i + L\sum_i \frac{C_i}{T_i} \leq \sum_i C_i + cL \right) \Rightarrow \left(L \leq \frac{\sum_i C_i}{1-c} \right) \qquad (3)$$

Comme $\sum_i C_i \le \sum_i \frac{C_i}{T_i} \cdot \max(T_i) \le c \cdot \max(T_i)$, nous avons le majorant suivant sur le nombre d'itérations:

$$\sum_i \lceil L/T_i \rceil \le n \left\lceil \frac{c}{(1-c)} \cdot \delta T \right\rceil .$$

A chaque itération, le nombre d'opérations élémentaires pour résoudre l'équation $L^{(m+1)} = \sum_i \left\lceil L^{(m)}/T_i \right\rceil C_i$ et la comparer à la valeur de $L^{(m)}$ est égal à $4n\xi$ (plus précisément *n-1* additions, *n* divisions, *n* multiplications, *n* parties supérieures et une comparaison).
Le coût du calcul de L est donc majoré par $4n^2 \left\lceil \frac{c}{(1-c)} \cdot \delta T \right\rceil \xi$. ❒

III.2.2. Période occupée de priorité maximale

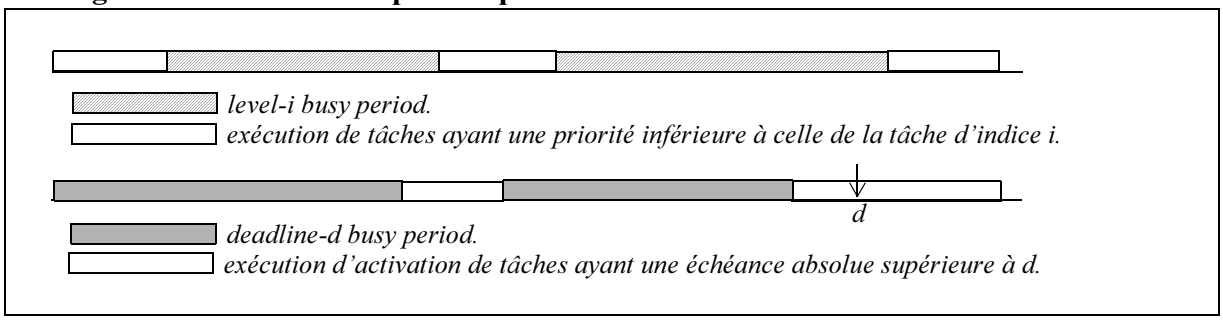
Il est possible de spécialiser le concept de période occupée en le rapportant au type de priorité utilisé (cf. Figure 3). Nous en conservons volontairement les appellations anglophones usuelles et concises. Dans le cas des priorités fixes, on parle de *level-i busy period.*

Définition 18 - *[Lehoczky 1990]. Une level-i busy period est une période occupée du processeur durant laquelle seules des activations de tâches ayant une priorité supérieure ou égale à τ_i (la tâche d'indice i) sont exécutées.*

De façon similaire, nous introduisons dans [George et al. 1996] le concept de *deadline-d busy period* qui fait référence à une échéance absolue, c.à.d. une priorité dynamique.

Définition 19 - *[George et al. 1996]. Une deadline-d busy period est une période occupée du processeur durant laquelle seules des activations de tâches ayant une échéance absolue inférieure ou égale à l'instant d sont exécutées.*

Figure 3 : Périodes occupées de priorité maximale



Les périodes occupées de priorités maximales font donc référence à un type de priorité. Contrairement aux périodes occupées du processeur, elles sont donc conditionnées par l'algorithme d'ordonnancement utilisé et pas uniquement par le scénario d'activation. La Figure 3 peut ainsi correspondre au même scénario d'activation (et donc à la même période occupée du processeur, cf. chapitre III.2.1. p. 17) mais la disposition interne des périodes occupées de priorités maximales diffère en fonction de l'algorithme d'ordonnancement utilisé.

Les outils suivants, basés sur de telles périodes occupées de priorités maximales, seront développés dans la suite, en fonction des algorithmes d'ordonnancement étudiés, pour :

- identifier les scenarii d'activation pires cas et tester la faisabilité de trafics non-concrets τ,
- effectuer les calculs de pires temps de réponse de toute tâche τ_i de τ,
- borner les intervalles d'études par tâche τ_i.

Définition 20 - *Par tâche* τ_i *:*

- $w_{i,q}$ *est la longueur de la level-i busy period conduisant à* $r_{i,q}$*, le pire temps de réponse de la* $(q+1)^{ième}$ *activations de la tâche* τ_i.
- $L_i(a)$ *est la longueur de la deadline*$(a+D_i)$ *busy period conduisant à* $r_i(a)$*, le pire temps de réponse de l'activation de la tâche* τ_i *arrivée à l'instant a.*
- L_i *est une borne sur* $w_{i,q}$*, ou* $L_i(a)$*, au delà de laquelle il n'est plus possible d'obtenir* r_i*, le pire temps de réponse de la tâche* τ_i*. Du Lemme 2 p. 17, nous savons que* $L_i \leq L$.

Nous identifierons lors de la partie B les algorithmes d'ordonnancement préemptifs qui utilisent ces notions et verrons comment les calculer. La partie C adaptera ces résultats en contexte non-préemptif non-oisif.

III.3. Référentiel d'ordonnancement

Au-delà des concepts classiques en ordonnancement temps réel, nous précisons maintenant notre cahier des charges. En effet, comme indiqué précédemment, les concepts usuels de faisabilité (cf. Définition 4 p. 10) et d'optimalité (cf. Définition 6 p. 11) peuvent être ambigus.

A titre d'exemple, une lecture rapide des résultats peut faire apparaître que des algorithmes aussi différents que *Earliest Deadline First* (*EDF*) et *Deadline Monotonic* (*DM*) sont optimaux sans pouvoir les départager. Comme nous le verrons lors de la partie B, *EDF* et *DM* sont bien optimaux pour des trafics non-concrets vérifiant $T_i \geq D_i$, $\forall i \in [1, n]$, mais parmi un ensemble plus restreint de solutions algorithmiques possibles pour *DM*.

Une lecture plus attentive des résultats permet bien sur de faire la différence. Cependant, afin de lever toute ambiguïté sur l'ensemble des résultats énoncés, nous utiliserons le concept de référentiel d'ordonnancement introduit dans [Hermant et al. 1996] par notre co-auteur L. Leboucher ainsi que quelques définitions résultantes[1]. Celles-ci sont abstraites pour l'instant mais prendront leur intérêt lors des parties suivantes. Nous précisons juste la définition de la faisabilité retenue et la discuterons lors du chapitre III.3.2. p. 22.

III.3.1. Concepts de base

Définition 21 - *[Hermant et al. 1996], [Leboucher 1998]. Un* ***Référentiel d'ordonnancement*** Σ *est un couple* (Υ,Π) *où* Υ *est un ensemble de trafics et* Π *un ensemble d'algorithmes d'ordonnancement.*

En d'autres termes, Υ représente un problème posé en termes de trafics (par exemple $\forall(\tau \in \Upsilon)$, $\forall i \in [1, n]$, $T_i = D_i$) et Π les solutions algorithmiques envisagées (par exemple uniquement les algorithmes préemptifs à priorités fixes).

Les référentiels d'ordonnancement permettent aussi de définir rigoureusement les propriétés "temps réel" que l'on cherche à vérifier dans Σ, la faisabilité en particulier qui sert de référence aux autres propriétés.

Nous attirons l'attention du lecteur sur le fait que plusieurs définitions de la faisabilité d'un trafic sont possibles dans Σ et que la validité des résultats obtenus en dépend. Nous précisons ici la définition implicitement utilisée dans [Hermant et al. 1996] et retenue pour la suite de ce travail. Nous discuterons brièvement lors du chapitre VI.1.2. p. 52, de l'impact d'autres définitions possibles de la faisabilité dans Σ.

1. Elles sont développées dans [Leboucher 1998].

Définition 22 - *Soit* $\Sigma=(\Upsilon,\Pi)$ *un référentiel d'ordonnancement. Soit* $Q \in \Pi$ *et* $\tau \in \Upsilon$ *un trafic non-concret,* $Q(\tau) \Leftrightarrow (\forall \omega \in \tau, Q(\omega))$ *signifie que tout trafic concret* ω *issu de* τ *vérifie* $Q(\omega)$*, c.à.d. est ordonnançable de façon valide par Q. On dit alors que* τ *est Q-faisable.*

Définition 23 - *Soit* $\Sigma=(\Upsilon,\Pi)$ *un référentiel d'ordonnancement. Soit* $\tau \in \Upsilon$ *un trafic non-concret,* τ *est dit* Σ***-faisable*** *si et seulement si:* $(\exists Q \in \Pi, Q(\tau))$.

Cette définition précise et remplace la Définition 4 p. 10. Notre cahier des charges considèrant des trafics non-concrets τ (cf. Hypothèse 1 p. 9), des algorithmes en-ligne, non-probabilistes (cf. Hypothèse 2 p. 10) et des tests de faisabilité les moins couteux possibles, la définition choisie ici impose que tout scénario d'activation ω d'un trafic non-concret τ soit ordonnançable par **un même** algorithme dans Π pour que τ soit déclaré faisable dans Σ.

La complexité des tests de faisabilité (cf. Propriété 4 p. 12) dépend par définition de l'énoncé de la faisabilité choisi. Voyons ce qu'il en est de l'optimalité (cf. Propriété 3 p. 11).

Définition 24 - *[Hermant et al. 1996], [Leboucher 1998]. Soit* $\Sigma=(\Upsilon,\Pi)$ *un référentiel d'ordonnancement.* $P \in \Pi$ *est dit* Σ***-optimal*** *si et seulement si:*

$$(\forall \tau \in \Upsilon), (\exists Q \in \Pi, Q(\tau)) \Rightarrow P(\tau).$$

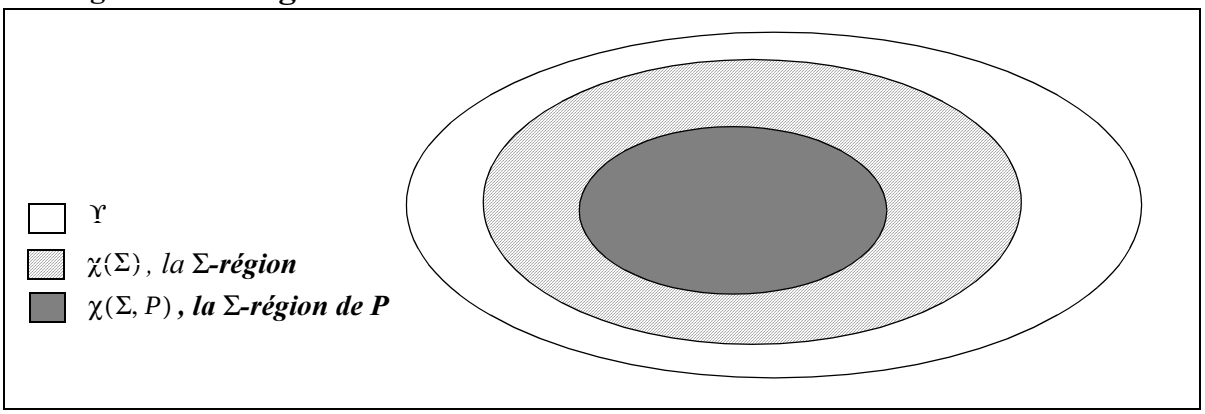
En d'autres termes, un algorithme Σ-optimal est un algorithme qui sait ordonnancer tout trafic non-concret τ Σ-faisable (c.à.d. ici tel que tous ses scénarii d'activation soient ordonnançables par un même algorithme dans Σ). Cette définition de la Σ-optimalité est bien conditionnée par la définition précédente de la Σ-faisabilité (sur le coté gauche de l'implication), elle remplace dorénavant la Définition 6 p. 11.

Nous pouvons maintenant faire explicitement référence à un ensemble d'algorithmes d'ordonnancement et un ensemble de trafics pour exprimer la notion de faisabilité et d'optimalité. L'ambiguïté est donc levée.

De nombreux autres concepts peuvent se déduire des référentiels d'ordonnancement (cf. [Leboucher 1998]). Nous précisons juste ici ceux qui seront cités ou utilisés dans la suite.

Définition 25 - *[Hermant et al. 1996], [Leboucher 1998]. Soit* $\Sigma=(\Upsilon,\Pi)$*, un référentiel d'ordonnancement. L'ensemble des trafics* Σ*-faisables,* $\chi(\Sigma) = \{\tau \in \Upsilon, (\exists P \in \Pi, P(\tau))\}$*, est appelé la* Σ***-région****. Similairement, l'ensemble des trafics P-faisables,* $\forall P \in \Pi$*,* $\chi(\Sigma, P) = \{\tau \in \Upsilon, P(\tau)\}$*, est appelé la* Σ***-région de P.***

Figure 4 : Σ-région



Au-delà de l'optimalité, la Propriété 3 p. 11, fait aussi référence à notion d'efficacité des algorithmes. Nous rappellerons comment L. Leboucher formalise cette notion lors du chapitre VI.1.2. p. 52. Voyons dans un premier temps comment il caractérise l'ensemble Υ pour les trafics non-concrets τ qui nous intéressent (cf. Hypothèse 1 p. 9).

Définition 26 - *[Leboucher 1997], [Leboucher 1998]. Soit τ et τ' deux trafics non-concrets et λ, un réel positif :*

- *le trafic non-concret constitué du produit cartésien[1] de l'ensemble des scénarii d'activation ω de τ avec l'ensemble des scénarii d'activation ω' de τ' est appelé l'**addition** de τ et τ'. Il est noté $\tau + \tau' = \{(\omega, \omega') / \omega \in \tau, \omega' \in \tau'\}$ (une loi de composition interne associative).*
- *similairement, le trafic non-concret constitué des tâches de τ avec une durée d'exécution multipliée par λ est appelé la **multiplication** de τ par λ (un opérateur associatif).*

Définition 27 - *[Hermant et al. 1996], [Leboucher 1998]. Une tâche périodique non-concrète est caractérisée par un couple (T,D) ou T est la période et D l'échéance relative de la tâche. Si τ est une addition de tâches périodiques non-concrètes, τ est appelé un Trafic Périodique Non-concret (**TPN**). Un **référentiel d'ordonnancement périodique** est alors un référentiel d'ordonnancement $\Sigma=(\Upsilon,\Pi)$ où Υ est un **ETPN**, c.à.d., un Ensemble de Trafics Périodiques Non-concrets.*

Nous utiliserons indifféremment les notations τ et TPN dans la suite. En effet, comme les trafics non-concrets qui nous intéressent sont caractérisés par une pire densité d'activation, les problèmes d'ordonnancement étudiés dans la suite pourront être modélisés par des référentiels d'ordonnancement où Υ est un ETPN.

Il doit être clair que Υ étant caractérisé par un ensemble de couples (T,D), tout TPN (c.à.d. tout trafic périodique non-concret τ) dont les tâches sont instanciées à partir de ces couples fait partie de Υ. Ceci **quelle que soit** la durée d'exécution des dites tâches et le nombre de tâches utilisant le même couple (T,D).

III.3.2. Discussion

Σ permet donc de poser les problèmes d'ordonnancement en termes (1) de trafics, (2) de solutions algorithmiques envisagées et (3) de propriétés à démontrer. Lors des parties suivantes, nous utiliserons la Définition 23 p. 21, de la Σ-faisabilité et la Définition 24 p. 21, de la Σ-optimalité pour proposer un panorama puis quelques extensions à l'ordonnancement temps réel "traditionnel". Ceci sera obtenu en variant les trafics et les solutions algorithmiques considérées dans Σ.

Σ ouvre cependant une perspective plus large en jouant aussi sur l'énoncé des propriétés[2]. Nous illustrons rapidement ceci sur les propriétés de faisabilité et d'optimalité. Notre but n'est pas de savoir si l'un des énoncés proposé (il y en a d'autres) est meilleur que les autres mais simplement d'attirer encore une fois l'attention du lecteur sur l'intérêt de Σ pour :

- fixer le domaine de validité d'un résultat d'ordonnancement
- couvrir différentes motivations applicatives (d'ailleurs plus ou moins pertinentes).

Rappelons (cf. Définition 22 p. 21) que $Q(\omega)$ signifie que le scénario d'activation ω est ordonnançable par l'algorithme Q et $Q(\tau) \Leftrightarrow (\forall \omega \in \tau, Q(\omega))$ signifie que tous les scénarii d'activation ω de τ sont ordonnançables par l'algorithme Q (on dit que τ est Q-faisable).

1. Il ne peut s'agir de l'union qui conduirait à $\tau+\tau=\tau$ et non 2τ comme attendu en préemptif (cf. chapitre VI.1.2. p. 52)
2. On imagine le nombre de problèmes possibles en variant les trafics, les solutions algorithmiques et les propriétés.

Il est possible pour commencer de s'en tenir aux trafics concrets ω (par exemple si l'on dispose d'un ensemble de scénarii d'activation sans liens apparents). La flexibilité offerte sur l'énoncé des propriétés est alors relativement faible.

Définition 28 - *Soit* $\Sigma=(\Upsilon,\Pi)$ *un référentiel d'ordonnancement. Soit* $\omega \in \Upsilon$ *un trafic concret,* ω *est dit* Σ***-faisable*** *si et seulement si:* $(\exists Q \in \Pi, Q(\omega))$.

Définition 29 - *Soit* $\Sigma=(\Upsilon,\Pi)$ *un référentiel d'ordonnancement.* $P \in \Pi$ *est dit* Σ***-optimal*** *si et seulement si:* $(\forall \omega \in \Upsilon), (\exists Q \in \Pi, Q(\omega)) \Rightarrow P(\omega)$.

Cette dernière définition impose que tout scénario d'activation ordonnançable dans Σ puisse être ramené à un ordonnancement valide par un algorithme Σ-optimal. Il est clair que trouver un tel algorithme dans un référentiel est intéressant mais aussi difficile. Nous verrons ainsi que, parmi les résultats énoncés dans la suite, seul l'algorithme *EDF* peut être systématiquement qualifié de Σ-optimal avec cette définition (cf. Théorème 1 p. 29).

Si nous revenons sur les trafics non-concrets τ, la flexibilité offerte sur l'énoncé des propriétés est plus importante. Voyons ainsi comment il est possible de relaxer progressivement la Définition 23 p. 21, de la faisabilité dans un même référentiel Σ.

Si le besoin applicatif est que tout scénario d'activation ω de τ soit ordonnançable par **au moins un** (et non par un même) algorithme dans Π pour que τ soit déclaré Σ-faisable, nous avons :

Définition 30 - *Soit* $\Sigma=(\Upsilon,\Pi)$ *un référentiel d'ordonnancement. Soit* $\tau \in \Upsilon$ *un trafic non-concret,* τ *est dit* Σ***-faisable*** *si et seulement si:* $\forall \omega \in \tau, (\exists Q \in \Pi, Q(\omega))$.

L'intérêt de cette définition est que l'ensemble des trafics Σ-faisables augmente en comparaison de la Définition 23 p. 21, puisque τ peut être dit Σ-faisable, même si un choix parmi un ensemble d'algorithmes est nécessaire pour ordonnancer correctement tous ses scénarii d'activation. Comme il parait naturel de vouloir éviter un tel choix en-ligne, il est alors important de trouver un algorithme Σ-optimal. L'inconvénient est alors que cette algorithme doit maintenant vérifier :

Définition 31 - *Soit* $\Sigma=(\Upsilon,\Pi)$ *un référentiel d'ordonnancement.* $P \in \Pi$ *est dit* Σ***-optimal*** *si et seulement si:* $(\forall \tau \in \Upsilon), (\forall \omega \in \tau, (\exists Q \in \Pi, Q(\omega))) \Rightarrow P(\tau)$.

La Σ-faisabilité intervient dans la partie gauche de l'implication utilisée pour énoncer la Σ-optimalité. Relaxer la Σ-faisabilité conduit donc à durcir la Σ-optimalité qui s'énonçait auparavant : $(\forall \tau \in \Upsilon), (\exists Q \in \Pi, Q(\tau)) \Rightarrow P(\tau)$, par la Définition 24 p. 21.

Si le besoin applicatif est qu'**au moins** une instanciation concrète ω de τ soit ordonnançable par **un** algorithme dans Π pour que τ soit déclaré Σ-faisable, nous avons :

Définition 32 - *Soit* $\Sigma=(\Upsilon,\Pi)$ *un référentiel d'ordonnancement. Soit* $\tau \in \Upsilon$ *un trafic non-concret,* τ *est dit* Σ***-faisable*** *si et seulement si:* $\exists \omega \in \tau, (\exists Q \in \Pi, Q(\omega))$.

L'intérêt est ici encore d'augmenter l'ensemble des trafics non-concrets Σ-faisables. L'utilisation pratique d'une telle définition ne pourra être la même cependant puisque seule une instanciation concrète ω de τ sera ordonnançable à coup sur si τ est Σ-faisable. Il faudra, par exemple, l'identifier lors d'une analyse préalable ayant pour objectif de fixer le scénario d'activation opérationnel ω par trafic non-concret τ. Malheureusement, la Σ-optimalité va encore être rendue plus difficile à établir.

Définition 33 - *Soit* $\Sigma=(\Upsilon,\Pi)$ *un référentiel d'ordonnancement.* $P \in \Pi$ *est dit* Σ***-optimal*** *si et seulement si:* $(\forall \tau \in \Upsilon), (\exists \omega \in \tau, (\exists Q \in \Pi, Q(\omega))) \Rightarrow P(\tau)$.

Cette dernière définition de la Σ-optimalité est en effet très restrictive puisque P doit ordonnancer tous les scénarii d'activation ω de τ dès lors qu'un algorithme dans Σ sait ordonnancer l'un d'entre eux.

Si l'on souhaite être moins restrictif, il est possible d'adopter la définition suivante qui signifie que P doit ordonnancer au moins un scénario d'activation ω de τ dès lors qu'un algorithme dans Σ sait en ordonnancer au moins un.

Définition 34 - *Soit* $\Sigma=(\Upsilon,\Pi)$ *un référentiel d'ordonnancement.* $P \in \Pi$ *est dit* ***Σ-optimal*** *si et seulement si:* $(\forall \tau \in \Upsilon), (\exists \omega \in \tau, (\exists Q \in \Pi, Q(\omega))) \Rightarrow (\exists \omega' \in \tau, P(\omega'))$.

Notons que cette dernière définition modifie l'approche précédente puisqu'en plus de repercuter la Σ-faisabilité sur la partie gauche de l'implication, nous avons aussi relaxé à droite la condition à vérifier par l'algorithme Σ-optimal.

En conclusion, on voit bien sur ces exemples qu'il est possible de varier à volonté la définition des proprités que l'on souhaite vérifier dans Σ mais que la motivation applicative sous-jacente est fondamentale. D'autres propriétés existent (dominance, efficacité...) pour lesquelles il serait aussi possible de varier les définitions. Une question ouverte par ces quelques exemples est de systématiser les référentiels Σ et les définitions de propriétés pour lesquels les résultats d'ordonnancement présentés dans la suite s'appliquent.

Partie B : Ordonnancement temps réel centralisé préemptif

Cette partie est consacrée à l'ordonnancement centralisé préemptif en présence de trafics non-concrets τ. Depuis [Liu and Layland 1973], la publication de référence en ordonnancement temps réel, ce problème a été largement étudié. Deux approches existent mais n'offrent pas vraiment de points de comparaison en termes d'outils et de performances. La première se base sur les priorités fixes, simples à implémenter, alors que la seconde s'intéresse aux priorité dynamiques, théoriquement supérieures (dont les résultats seraient plus optimaux en quelque sorte !!!). L'objet de cette partie est de donner quelques éléments de comparaison entre ces deux points de vue.

Le chapitre IV. p. 28, commence par homogénéiser l'énoncé des résultats connus en termes d'optimalité, de faisabilité et de pires temps de réponse. Ces résultats sont assez complets mais quelques extensions/discussions sont toutefois proposées au chapitre V. p. 41.

Le chapitre VI. p. 51, donne ensuite quelques éléments de comparaison en termes d'efficacité des algorithmes d'ordonnancement ainsi que du coût des tests de faisabilité associés. Une synthèse sera proposée lors de la Partie D au regard des propriétés posées dans le cahier des charges, et illustrée par quelques applications numériques.

Les résultats présentés dans cette partie sont en grande partie issus des deux rapports [Hermant et al. 1996] et [George et al. 1996].

IV. Etat de l'art

IV.1. Introduction

Le principal objectif des travaux existants est de coupler un algorithme d'ordonnancement optimal avec un test nécéssaire et suffisant pseudo-polynomial, basé ou non sur le calcul des pires temps de réponse des tâches. L'ordonnancement temps réel centralisé préemptif a fait l'objet de nombreux travaux. La Table 3 qui suit ne doit pas être considérée comme une liste exhaustive des publications dans ce domaine mais davantage comme une synthèse des principaux résultats en notre connaissance.

Table 3 : Principaux résultats préemptifs

	Priorités dynamiques, cf. chapitre IV.2. p. 28	Priorités fixes, cf. chapitre IV.3. p. 35
Optimalité	[Liu and Layland 1973], [Dertouzos 1974], [Mok 1983]	[Liu and Layland 1973], [Leung and Whitehead 1982], [Audsley 1991]
test donnant la faisabilité seule	[Liu and Layland 1973], [Leung and Merril 1980], [Baruah et al. 1990],[Baruah et al. 1990b], [Katcher et al. 1993], [Shin and Zheng 1994], [Ripoll et al. 1996]	(conditions suffisantes uniquement) [Liu and Layland 1973], [Lehoczky 1990]
test donnant la faisabilité et les pires temps de réponse	[Spuri 1996]	[Leung and Whitehead 1982], [Joseph and Pandya 1986], [Lehoczky 1990], [Burns et al. 1994]

Les algorithmes d'ordonnancement temps réel considérés sont *HPF*, préemptifs non-oisifs et se distinguent par leurs techniques d'assignation de priorités (cf. Définition 8 p. 13). Nous examinons ici les résultats connus, d'abord en présence de priorités dynamiques (cf. chapitre IV.2. p. 28) puis de priorités fixes (cf. chapitre IV.3. p. 35). Notons que :

- nous suivrons l'ordre dans lequel ont été introduits les outils et les résultats en relaxant progressivement les relations particulières entre les paramètres des tâches. Nous utiliserons cependant systématiquement le concept de référentiel d'ordonnancement (cf. Definition 21 p. 20) afin de lever toute ambiguïté sur les énoncés. Les cases grisées sont celles pour lesquelles quelques extensions/discussions sont proposés lors du chapitre V. p. 41.
- notre cahier des charges ne prend pas en compte les problèmes d'implantation physique. le lecteur intéressé trouvera des informations sur ces sujets dans [Yuan 1991] et [Spuri 1995] pour *EDF* et dans [Audsley 1995] pour les priorités fixes.

IV.2. Priorités dynamiques

IV.2.1. Généralités

Les algorithmes d'ordonnancement à priorités dynamiques sont caractérisés par une assignation des priorités qui procède en-ligne par activation de tâches (cf. Définition 8 p. 13), et non par tâche, hors-ligne, comme c'est le cas des algorithmes d'ordonnancement à priorités fixes. Ainsi, l'algorithme préemptif, non-oisif, *Earliest Deadline First* (noté ***EDF*** dans la suite), assigne dynamiquement les priorités aux activations de tâches en fonction de leurs échéances absolues. A

tout instant, l'activation de tâche en file d'attente ayant l'échéance absolue la plus proche se voit donc assigner la plus haute priorité par *EDF*. Notons que :

- *EDF* conduit à un résultat d'optimalité très général qui se traduira par une efficacité maximale lors du chapitre VI.1. p. 51.
- dans l'état de l'art, contrairement aux algorithmes à priorités fixes, l'établissement d'un test nécéssaire et suffisant pour la faisabilité d'un trafic par *EDF* n'est pas nécessairement lié au calcul des pires temps de réponses des tâches. Nous séparerons donc ici l'exposé des résultats traitant de la faisabilité seule de ceux traitant des pires temps de réponse.

IV.2.2. Optimalité

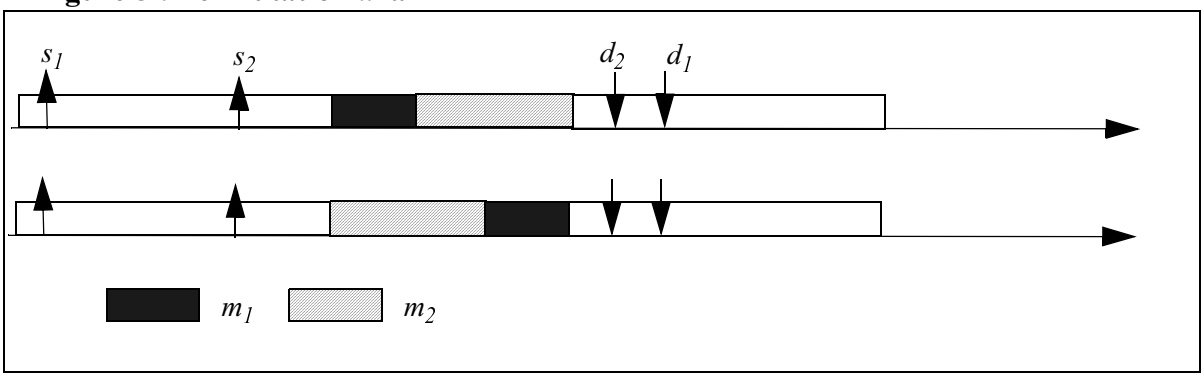
Le résultat suivant s'applique à un référentiel d'ordonnancement très général en termes de trafics et d'algorithmes d'ordonnancement.

Théorème 1 - *[Dertouzos 1974]. EDF est* Σ*-optimal dans un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout trafic (concret ou non-concret) et* Π *contient tout algorithme d'ordonnancement (préemptif ou non-préemptif, oisif ou non-oisif, à priorités fixes ou dynamiques).*

La preuve de Dertouzos montre qu'il est toujours possible, par permutation, de transformer un ordonnancement valide quelconque en un ordonnancement *EDF* valide. Autrement dit quelque soit soit le scénario d'activation ω ordonnançable, alors ω est ordonnançable par *EDF* (ce que nous notons *EDF*(ω)). Ce résultat de Σ-optimalité très fort (cf. Définition 29 p. 23, pour une discussion) est donc valable pour notre définition de la Σ-optimalité (cf. Définition 24 p. 21) qui traite de trafics non-concrets τ et qui s'énonce ici de la façon suivante : $(\forall \tau \in \Upsilon), (\exists Q \in \Pi, Q(\tau)) \Rightarrow EDF(\tau)$ où $Q(\tau) \Leftrightarrow (\forall \omega \in \tau, Q(\omega))$ et $EDF(\tau) \Leftrightarrow (\forall \omega \in \tau, EDF(\omega))$.

Rappelons le principe (très simple) de cette preuve en partant de m_1 et m_2, les durées d'exécution consécutive de deux portions de tâche dans un ordonnancement initial valide (cf. Figure 5). Si les instants d'activations de m_1 et m_2 (notés s_1 et s_2) précèdent l'exécution de m_1 alors le travail correspondant à m_1 et m_2 a séjourné simultanément en file d'attente avant d'être exécutés. Si de plus l'échéance absolue de m_1 (notée d_1) est supérieure à celle de m_2 (notée d_2) alors l'ordre d'exécution ne respecte pas *EDF*. En permutant "à la *EDF*" m_1 et m_2, le reste de l'ordonnancement initial n'est pas modifié car $m_1+m_2=m_2+m_1$. Pour la même raison les échéances absolues d_1 et d_2 sont toujours respectées si par hypothèse l'ordonnancement initial est valide. En réitèrant ce raisonnement, l'ordonnancement final est bien celui qu'aurait généré *EDF*.

Figure 5 : Permutation à la *EDF*



Notons que l'algorithme *Least Laxity First* (noté *LLF*, cf. [Mok 1983]), qui a tout instant exécute l'activation de tâche ayant la plus petite laxité[1], a aussi été montré optimal dans certains contextes. Il est cependant plus complexe à mettre en oeuvre puisqu'il nécessite de connaitre à tout instant la durée d'exécution restante par activation de tâche présente en file d'attente. *LLF* conduit

1. échéance absolue moins temps courant moins durée d'exécution restante.

aussi à plus de préemption que *EDF* et à de mauvais temps de réponse. Par exemple, si deux activations de tâches ont la même échéance absolue : avec *EDF* l'une d'entre elles s'exécutera prioritairement et sans préemption alors qu'avec *LLF* ces deux activations se préempteront et se retarderont mutuellement tant qu'aucune des deux n'aura pas fini de s'exécuter. Comme enfin les résultats connus en termes de faisabilité avec *LLF* sont sans comparaison avec ceux établis pour *EDF*, nous focaliserons sur ce dernier algorithme même si nous sommes conscient qu'une justification plus formelle est à faire.

IV.2.3. Faisabilité seule

IV.2.3.1. Scénario d'activation pire cas et demande processeur

Un résultat classique de [Liu and Layland 1973] montre que le scénario d'activation synchrone (cf. Définition 11 p. 15) est le pire en terme de faisabilité pour tout trafic ordonnancé par *EDF*. Nous en donnons l'énoncé de [Spuri 1996] plus informatif. [Katcher et al. 1993] puis [Ripoll et al. 1996] l'établissent préalablement en présence de trafics plus contraints.

Lemme 5 - *[Liu and Layland 1973], [Katcher et al. 1993], [Ripoll et al. 1996], [Spuri 1996]. Tout trafic non-concret τ ordonnancé par EDF est faisable si et seulement si aucune activation de tâche ne rate son échéance durant la période occupée synchrone du processeur.*

La preuve montre simplement que si ω, un scénario d'activation quelconque de τ, conduit à rater une échéance absolue avec *EDF*, alors le scénario d'activation synchrone conduit aussi à rater une échéance. Nous renvoyons le lecteur au Lemme 9 p. 41, pour une démonstration et un raffinement de ce résultat grâce au concept de *deadline busy period*.

Une conséquence immédiate est qu'une procédure simple, pour établir la faisabilité de tout trafic non-concret τ, consiste à dérouler l'ordonnancement *EDF* sur la période occupée synchrone du processeur (cf. Définition 17 p. 17) et à vérifier que les échéances absolues y sont respectées. Rappelons que la complexité du calcul de L, la taille de cette période occupée, est pseudo-polynomiale lorsque la charge $U \leq c < 1$ (cf. Lemme 4 p. 18).

Au-delà des approches procédurales et autres résultats historiques, que nous évoquons brièvement, l'approche actuellement utilisée pour tester la faisabilité des trafics avec *EDF* est basée sur une évaluation de la demande processeur $h(t)$ sur des intervalles bornés (cf. Définition 15 p. 16). Rappelons que $h(t)$ représente en tout $t \geq 0$, avec un instant synchrone en 0 (cf. Définition 11 p. 15), la quantité maximale des durées d'exécution d'activations de tâches de τ arrivées dans l'intervalle $[0, t]$ et dont les échéances absolues sont aussi dans cet intervalle. Ces activations sont donc exécutées en priorité par *EDF*.

Une condition nécessaire pour établir la faisabilité d'un trafic non-concret τ par n'importe quel algorithme est de vérifier que $\forall t \geq 0,\ h(t) \leq t$ (cf. Lemme 1 p. 16). Examinons maintenant comment cette approche a permis d'obtenir des conditions suffisantes avec *EDF*, sous différentes hypothèses sur les paramètres de τ. Notons que les tests nécéssaires et suffisants obtenus sont autant de preuves de la Σ-optimalité de *EDF* en présence de trafics non-concret τ dans des référentiels particuliers.

IV.2.3.2. Le cas $D_i \geq T_i$, $\forall i \in [1, n]$

Le résultat suivant généralise le test nécéssaire et suffisant initialement établi par [Liu and Layland 1973] dans le cas $D_i = T_i$, $\forall i \in [1, n]$.

Théorème 2 - *[Liu and Layland 1973], [Baruah et al. 1990b]. Soit un référentiel d'ordonnancement $\Sigma = (\Upsilon, \Pi)$ où Υ contient tout TPN caractérisé par $D_i \geq T_i,\ \forall i \in [1, n]$ et Π contient tout algorithme d'ordonnancement. EDF est Σ-optimal et $\forall (\tau \in \Upsilon)$, τ est EDF-faisable si et seulement si:*

$$U \leq 1. \qquad (4)$$

Un TPN est un Trafic Périodique Non-concret (cf. Définition 27 p. 22). L'intérêt de ce résultat, basé sur la charge (cf. Définition 9 p. 15), est d'établir simplement la faisabilité d'un TPN par *EDF*. Ce test est trivialement nécessaire puisque pour ce type de trafic, la charge de travail ne peut être supérieure à la capacité de traitement pour aucun algorithme. [Baruah et al. 1990b] montrent qu'elle est aussi suffisante pour *EDF* (qui est donc Σ-optimal) grâce au concept de demande processeur. En effet, si l'on suppose que $\forall t \geq 0$, $h(t) \leq t$, comme par hypothèse $D_i \geq T_i$, $\forall i \in [1, n]$, nous avons :

$$\forall t \geq 0, \qquad h(t) = \sum_{D_i \leq t}(1 + \lfloor (t - D_i)/T_i \rfloor)C_i \leq t\sum_{i=1}^{n}\frac{C_i}{T_i} \leq t, \text{ c.à.d. } U \leq 1.$$

IV.2.3.3. Le cas $D_i \leq T_i$, $\forall i \in [1, n]$

Dans ce cas il ne peut y avoir simultanément plus d'une activation par tâche τ_i en file d'attente pour que τ soit faisable. En effet, sous l'hypothèse $D_i \leq T_i$, $\forall i \in [1, n]$, la plus ancienne activation de τ_i présente en file d'attente aurait alors nécessairement ratée son échéance absolue.

Historiquement, un test nécéssaire et suffisant fut proposé par [Leung and Merril 1980] en présence de trafics périodiques **concrets** ω, c.à.d. avec des instants d'activations connus et non nécessairement synchrones. Ils montrent qu'après une période transitoire tout ordonnancement non-oisif devient cyclique de période P, le ppcm des périodes des tâches (cf. Définition 13 p. 15). Ainsi la faisabilité de ω peut alors être vérifiée en déroulant l'ordonnancement *EDF* sur l'intervalle $[0, 2P + max\{s_i\}]$, avec s_i l'instant de première activation de la tâche ω_i et $min\{s_i\} = 0$ (cf. Définition 1 p. 8). Il suffit alors de vérifier que:

- (C1) aucune échéance absolue n'est ratée dans cet intervalle,
- (C2) la configuration[1] coïncide aux instants $P + max\{s_i\}$ et $2P + max\{s_i\}$.

Cette condition de faisabilité est procédurale puisqu'on déroule *EDF* sur un intervalle borné. Elle est améliorée par [Baruah et al. 1990] qui montrent que (C2) est toujours respectée si $U \leq 1$. (C1) s'opère cependant en un temps exponentiel puisque P est en pire cas fonction du produit des périodes des tâches. Notons d'une part que cette complexité ne cadre pas avec notre cahier des charges (cf. Propriété 4 p. 12) et d'autre part, que les trafics concrets considérés par cette procédure ne sont pas ceux qui nous intéressent (cf. Hypothèse 1 p. 9).

Le problème de savoir si un trafic périodique concret ω, à priori non synchrone, peut conduire à un instant synchrone est un problème NP complet [Leung and Merril 1980]. Avec les trafics non-concrets τ par contre, cette information est connue puisque le scénario d'activation synchrone est un des scenarii possibles (cf. Définition 2 p. 9). L'intervalle d'étude de (C1) est alors réduit à $[0, P]$ puisqu'alors $max\{s_i\} = 0$. Il n'y a plus de période transitoire.

Il est alors intéressant de faire le rapprochement avec le résultat du Lemme 5 p. 30 qui aboutit au même pire scénario d'activation périodique, synchrone, en terme de faisabilité pour un trafic non-concret τ. Comme ce dernier résultat limite l'intervalle d'étude à $[0, L]$ avec $0 \leq L \leq P$ lorsque $U \leq c < 1$ (cf. Lemme 3 p. 18), il est dominant.

Voyons maintenant comment la demande processeur conduit à un test nécéssaire et suffisant, non procédural, pour les trafics non-concrets. Tout d'abord, comme les échéances relatives sont inférieures ou égales aux périodes (par hypothèse que $D_i \leq T_i$, $\forall i \in [1, n]$), nous avons :

$$h(t) = \sum_{i=1}^{n} max(0, 1 + \lfloor (t - D_i)/T_i \rfloor)C_i = \sum_{i=1}^{n}(1 + \lfloor (t - D_i)/T_i \rfloor)C_i.$$

1. définie par Leung et Merril comme la quantité de travail déjà exécutée par tâche depuis sa dernière activation

Théorème 3 - *[Baruah et al. 1990], [Baruah et al. 1990b], [Ripoll et al. 1996]. Soit un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN caractérisé par* $D_i \leq T_i, \ \forall i \in [1, n]$ *et* Π *contient tout algorithme d'ordonnancement. EDF est* Σ*-optimal et* $\forall(\tau \in \Upsilon)$ *et* $U < 1$, τ *est EDF-faisable si et seulement si:*

$$\forall t \in [0, \sum_{i=1}^{n} (1 - D_i/T_i)C_i/(1-U)], \quad h(t) \leq t \tag{5}$$

Cette condition est trivialement nécessaire pour tout algorithme (cf. Lemme 1 p. 16). Par contradiction elle est aussi suffisante avec *EDF* (qui est donc Σ-optimal). Supposons en effet que l'éq. (5) soit vérifiée mais qu'il existe pourtant un scénario d'activation ω conduisant à un instant t_2 tel qu'une échéance absolue soit ratée avec *EDF*. Soit t_1 l'instant précédent t_2 tel que durant l'intervalle $[t_1, t_2]$, le processeur est continuellement occupé à exécuter des activations de tâches ayant des échéances absolues inférieures ou égales à t_2, c.à.d. exécutées en priorité par *EDF*.

Si nous appliquons le Lemme 5 p. 30 en ramenant le scénario d'activation initial à un scénario synchrone en t_1, alors la quantité de travail exécutée en priorité par *EDF* vaut toujours $h(t_2 - t_1) > t_2 - t_1$ (c.à.d. $h(t) > t$ si $t_1=0$ et $t_2=t$). Par manipulations algébriques :

$$t < h(t) \leq \sum_{i=1}^{n} (1 + (t - D_i)/T_i)C_i = tU + \sum_{i=1}^{n} (1 - D_i/T_i)C_i,$$

$$\text{c.à.d. } t < \sum_{i=1}^{n} (1 - D_i/T_i)C_i/(1-U) \text{ (contradiction avec l'éq. (5)).}$$

L'inconvénient de cette approche est que l'intervalle d'étude peut être grand si U est proche de 1. [Ripoll et al. 1996] proposent de combiner cette borne avec L, la taille de la période occupée du processeur, dont nous savons qu'elle limite aussi l'intervalle d'étude (cf. Lemme 5 p. 30). Ils montrent par des simulations que l'intérêt de ces deux bornes varie en fonction du trafic considéré. Le coût de ce test pseudo-polynomial sera examiné au chapitre VI.2. p. 61.

IV.2.3.4. Trafics généraux

Pour ces trafics non-concrets τ, on peut avoir plus d'une activation par tâche en file d'attente sans rater d'échéances. Il est en effet possible de réactiver une tâche sans que l'activation précédente ait fini d'être exécutée puisque $D_i \geq T_i$ pour certaines tâches (cf. Définition 2 p. 9). La formule de la demande processeur ne se simplifie donc pas comme dans le cas précédent.

Théorème 4 - *[Shin and Zheng 1994]. Soit un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN et* Π *contient tout algorithme d'ordonnancement. Avec* $U < 1$, *EDF est* Σ*-optimal et* $\forall(\tau \in \Upsilon)$, τ *est EDF-faisable si et seulement si:*

$$\forall t \in S, \quad h(t) \leq t \tag{6}$$

$$\text{où } S = (\bigcup_{i=1}^{n} \{kT_i + D_i, k \in \mathbb{N}\}) \cap \left[0, max\left\{max\{D_i\}, \sum_{i=1}^{n} (1 - D_i/T_i)C_i/1-U\right\}\right].$$

L'approche du Théorème 3 p. 32 s'adapte sans problèmes. Les améliorations/ modifications proposées pour prendre en compte les trafics généraux τ sont :

- de limiter l'évaluation de *h(t)* à l'ensemble *S* des échéances absolues générées par le scénario d'activation périodique synchrone. Seuls ces points sont en effet pertinents car ils correspondent aux instants où la valeur de *h(t)* (une fonction discontinue et croissante en *t*, cf. Définition 15 p. 16) est modifiée.
- de rajouter la borne $max\{D_i\}$. En effet, en faisant l'hypothèse que l'éq. (6) est vérifiée et qu'il existe pourtant un instant t où $h(t) > t$, alors les manipulations algébriques du théorème précédent ne sont valables que pour tout $t \geq max\{D_i\}$.

IV.2.4. Pires temps de réponse

Le résultat du Lemme 5 p. 30 définit le scénario pire cas pour la faisabilité d'un trafic non-concret τ par *EDF*. Par contre, il ne garantit pas que r_i, le pire temps de réponse d'une tâche τ_i de τ (cf. Définition 5 p. 10) sera obtenu par ce scénario. En fait, contrairement à l'intuition et à se qui se passe avec les algorithmes d'ordonnancement à priorités fixes, r_i résulte avec *EDF* d'un scénario critique (cf. Définition 10 p. 15) qui n'est pas nécessairement le scénario périodique synchrone. Ce problème, abordé depuis peu, débute par un résultat simple :

Lemme 6 - *[Spuri 1996]. Soit un trafic non-concret ordonnancé par EDF. Le scénario critique conduisant au pire temps de réponse d'une tâche τ_i se trouve dans une période occupée du processeur où toutes les autre tâches sont synchrones en début de cette période occupé et périodiques ensuite.*

Ce résultat n'établit pas le scénario critique exact de τ_i. Il utilise le raisonnement du Lemme 5 p. 30, pour montrer que les temps de réponse des activations de τ_i, présentes dans une période occupée du processeur, ne peuvent diminuer lorsque les autres tâches sont périodiques et synchrones en début de cette période occupée. Par contre, la priorité des activations de τ_i étant basées sur leurs échéances absolues, rien ne garantit que leurs temps de réponse augmenteront en déplaçant leurs instants d'arrivées, puisque leurs priorités sont alors modifiées. Nous renvoyons le lecteur au Lemme 10 p. 42 pour une démonstration et un raffinement de ce résultat grâce au concept de *deadline busy period.*

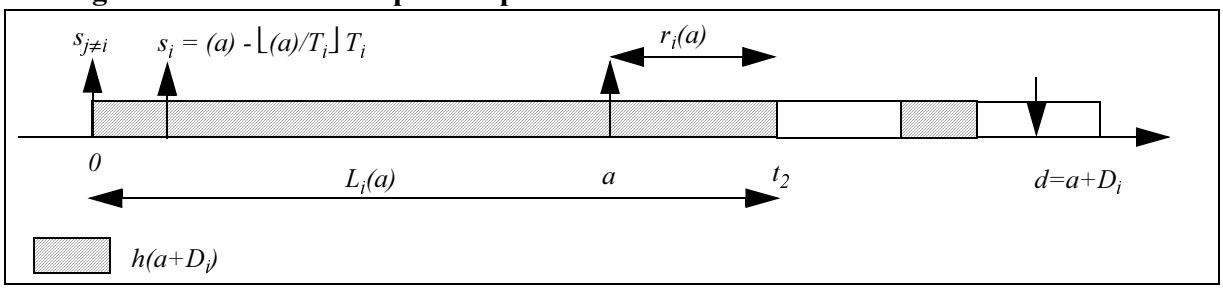
La solution proposée par [Spuri 1996] pour calculer r_i est alors de tester tous les temps de réponse résultant d'un placement particulier des activations de τ_i dans le scénario d'activation précédent. Plus précisément (cf. Figure 6) si ce scénario démarre en *0*, toutes les tâches autres que τ_i sont périodiques et synchrones en 0. Nous avons $s_j = 0, \ \forall j \neq i$. Si d est l'échéance absolue de l'une des activation de τ_i, nous en déduisons son instant d'activation $a=d-D_i$. Une valeur de a permet alors de construire un scénario d'activation complet, incluant les activations de τ_i arrivées en ..., $a-2T_i$, $a-T_i$, a, $a+T_i$..., candidat à être le scénario conduisant à r_i.

En pratique, étant donné une valeur du paramètre a, nous avons $s_i = a - \lfloor a/T_i \rfloor T_i$ et le temps de réponse de l'activation de τ_i arrivée à l'instant a est donné par l'équation suivante :

$$r_i(a) = L_i(a) - a, \tag{7}$$

où $L_i(a)$ représente en contexte préemptif la longueur de la période occupée entre l'instant *0* et l'instant t_2 de fin d'exécution par *EDF* de l'activation de τ_i arrivée à l'instant a. t_2 est donc le dernier instant où cette activation peut être préemptée par une activation de tâche présente en file d'attente et plus prioritaire, c.à.d. ayant une échéance absolue inférieure ou égale à d :

Figure 6 : Calcul de temps de réponse avec EDF



[Spuri 1996] remarque que $L_i(a) \leq h(a + D_i)$, où $h(a + D_i)$ représente la demande processeur des activations de tâches ayant une échéance absolue inférieure ou égale à l'activation de τ_i arrivée à l'instant a. En effet, certaines activations de tâches faisant partie de $h(a + D_i)$, mais arrivées tardivement (après l'instant t_2), sont en quelque sorte "doublées" dans l'ordonnancement *EDF* réel par l'activation de τ_i arrivée à l'instant a (cf. Figure 6).

Calculons $L_i(a)$. Il y a à coup sûr $1 + \lfloor a/T_i \rfloor$ activations de τ_i dans $L_i(a)$. Par contre, avec *EDF*, pour toute tâche $\tau_{j\neq i}$, il y a au plus $max(0, 1 + \lfloor (a + D_i - D_j)/T_j \rfloor)$ activations de τ_j dans

$L_i(a)$ car seules celles-ci ont une échéance absolue inférieure ou égale à $a + D_i$. Cependant, en tout t nous ne pouvons considérer que les activations qui sont arrivées dans l'intervalle $[0, t[$. Rappelons en effet que seules sont candidates à la préemption les activations de tâches arrivées avant la fin d'exécution de l'activation de τ_i arrivée à l'instant a. En conséquence, en tout t, le nombre d'activations de τ_j à prendre en compte vaut :

$$min\{\lceil t/T_j \rceil, max(0, 1 + \lfloor (a + D_i - D_j)/T_j \rfloor)\}. \quad (8)$$

Pour calculer $L_i(a)$, [Spuri 1996] adapte alors le calcul récursif de L, la plus longue période occupée du processeur, exposé lors du chapitre III.2.1. p. 17. Comme nous venons de le voir, il impose le nombre d'activations de τ_i à $1 + \lfloor a/T_i \rfloor$ et utilise de façon récursive l'éq. (8) pour déterminer le nombre exact d'activations des autres tâches dans $L_i(a)$. Nous obtenons alors:

$$\begin{cases} L_i^{(0)}(a) = 0 \\ L_i^{(m+1)}(a) = W_i(a, L_i^{(m)}(a)) + (1 + \lfloor a/T_i \rfloor)C_i \end{cases} \quad (9)$$

$$\text{avec } W_i(a, t) = \sum_{j \neq i,\, D_j \leq a + D_i} min\{\lceil t/T_j \rceil, 1 + \lfloor (a + D_i - D_j)/T_j \rfloor\}C_j.$$

Pour tout $a \geq 0$, la récursion s'arrête alors lorsque deux valeurs consécutives $L_i^{(m)}(a)$ et $L_i^{(m+1)}(a)$ sont égales. $L_i(a)$ est alors égal à $L_i^{(m)}(a)$. L'éq. (9) est convergente si $U \leq 1$ car $L_i(a)$ est croissante en a (cf. Lemme 12 p. 44) et $L_i(a) \leq L$, avec L bornée sous l'hypothèse $U \leq 1$ (cf. Lemme 2 p. 17 et Lemme 3 p. 18). Notons que:

- de cette façon nous trouvons la plus petite racine de l'équation:

$$t = W_i(a, t) + (1 + \lfloor a/T_i \rfloor)C_i. \quad (10)$$

- ce calcul n'a d'intérêt que pour les valeurs de a tel que $L_i(a) > a + C_i$ sinon il n'y a pas assez d'activations de tâches prioritaires pour atteindre a et le scénario proposé ne peut donc conduire au pire temps de réponse. Afin de ne pas considérer les valeurs aberrantes, [Spuri 1996] propose de remplacer l'éq. (7) par $r_i(a) = \max\{C_i, L_i(a) - a\}$.

- [Spuri 1996] limite aussi les valeurs de a à l'intervalle $[0, L]$. Au delà, nous trouvons par définition une période inoccupée du processeur et le calcul doit alors être réinitialisé sur la nouvelle période occupée dont le scénario d'activation ne peut être pire d'après le Lemme 6 p. 33. Nous verrons au chapitre V.1.1. p. 41 qu'il est possible de limiter cet intervalle à $L_i \leq L$ (par tâche τ_i) grâce au concept de *deadline-d busy period* (cf. Définition 20 p. 20).

- $L_i(a)$ est une fonction discontinue croissante en a. Il est donc possible de se limiter (comme pour la faisabilité, cf. Théorème 4 p. 32) aux points de discontinuité en "calant" $a + D_i$ sur les échéances absolues générées par le scénario d'activation du Lemme 6 p. 33, c.à.d. à l'ensemble $a \in A \cap [0, L]$, où :

$$A = \bigcup_{j=1}^{n} \{kT_j + D_j - D_i, k \in \mathbb{N}\}.$$

Finalement un trafic non-concret τ sera faisable par *EDF* si et seulement si:

$$\forall i \in [1, n], \quad r_i = max_{a \in A}\{r_i(a)\} \leq D_i \quad . \quad (11)$$

Similairement à l'établissement de la faisabilité, une étude du coût, pseudo-polynomial lorsque $U \leq c < 1$, de calcul des pires temps de réponse sera proposée au chapitre VI.2. p. 61.

IV.3. Priorités fixes

IV.3.1. Généralités

Les algorithmes d'ordonnancement à priorités fixes sont caractérisés par une politique d'assignation de priorités hors-ligne, valable pour toutes les activations d'une même tâche (cf. Définition 8 p. 13). Les résultats présentés ici sont strictement limités à l'ensemble des algorithmes préemptifs, non-oisifs à priorités fixes. Nous verrons qu'il existe plusieurs techniques d'attribution de priorités optimales possibles en fonction de l'ensemble des trafics non-concrets considérés. Ces restrictions sur la définition de l'optimalité reflètent la supériorité théorique de *EDF*. Ainsi, un test nécéssaire et suffisant pour un algorithme à priorités fixes n'est que suffisant pour *EDF*. De même, un un test nécéssaire et suffisant pour *EDF* n'est que nécessaire avec un algorithme à priorités fixes "optimal" dans le même contexte. Ceci sera discuté plus en détail lors du chapitre VI.1. p. 51. Nous verrons notamment que la supériorité théorique (très relative en fonction de l'ensemble de trafic considérée) de *EDF* peut être évaluée grâce au concept d'efficacité. Pour l'instant, nous nous contentons d'utiliser le concept de référentiel d'ordonnancement (cf. chapitre III.3. p. 20) afin de lever toute ambiguïté sur l'énoncé des résultats de faisabilité et d'optimalité. Notons que :

- dans la littérature (et contrairement à *EDF*) l'établissement de la faisabilité d'un trafic non-concret τ pour les priorités fixes est étroitement lié au calcul des pires temps de réponses des tâches. Les résultats existants traitent donc ce point par la Propriété 2 p. 11 de notre cahier des charges et non simplement par la Propriété 1 p. 8.
- le lecteur intéressé par un état de l'art sur les algorithmes d'ordonnancement à priorité fixes dans ce contexte pourra consulter [Audsley 1995].
- contrairement à *EDF*, l'optimalité des algorithmes à priorités fixes est très dépendante du référentiel d'ordonnancement et plus particulièrement de l'énoncé de faisabilité choisi (cf. chapitre III.3.2. p. 22).

IV.3.2. Le cas $T_i = D_i, \forall i \in [1, n]$

IV.3.2.1. Scénario d'activation pires cas

Lemme 7 - *[Liu and Layland 1973]. Soit PF, un algorithme d'ordonnancement préemptif, non-oisif à priorités fixes et τ, un trafic non-concret caractérisé par $T_i = D_i, \forall i \in [1, n]$. Le scénario critique de toute tâche τ_i de τ est celui pour lequel τ_i est activée simultanément avec toutes les tâches de priorités supérieures et où la densité d'activation est périodique.*

La preuve montre que si un scénario d'activation ω de τ conduit à rater l'échéance d'une activation de la tâche τ_i en t, alors le scénario synchrone (cf. Définition 11 p. 15) en $t_0=t-T_i$ conduit, comme avec *EDF* (cf. Lemme 5 p. 30), à rater cette échéance. On ne s'intéresse cependant ici qu'aux instants d'activation des autres tâches et pas à leurs échéances. Il n'y a qu'une activation de τ_i en file d'attente dans l'intervalle $[t_0, t]$ (dans le cas contraire, une échéance serait nécessairement ratée avant t puisque $T_i = D_i$). Comme avec le scénario synchrone en t_0, le nombre d'activations des tâches de priorité supérieure est maximisé dans l'intervalle $[t_0,t]$ et que *PF* exécute en priorité ces activations, alors le temps de réponse de τ_i ne peut diminuer et conduit donc toujours τ_i à rater son échéance absolue en t.

En conséquence, le scénario synchrone est le scénario critique (cf. Définition 10 p. 15) donnant le pire temps de réponse de chaque tâche sur sa première activation. Il est de plus le pire en terme de faisabilité du trafic. Nous avons vu que ceci n'était vrai que pour l'analyse de faisabilité avec *EDF*.

IV.3.2.2. Optimalité

Dans ce cas le résultat suivant montre que l'algorithme d'ordonnancement préemptif, non-oisif à priorités fixes *HPF/Rate-Monotonic* (noté ***RM*** dans la suite) qui assigne les priorités en ordre inversement proportionnel aux périodes des tâches est optimal. Ainsi la tâche ayant la période la plus petite se verra attribuer la plus haute priorité (le tri est en $O(n log_2 n)$).

Théorème 5 - *[Liu and Layland 1973]. RM est* **Σ-***optimal dans un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN caractérisé par* $T_i = D_i, \forall i \in [1, n]$ *et* Π *contient tout algorithme d'ordonnancement préemptif, non-oisif à priorités fixes.*

La preuve montre que si un trafic non-concret $\tau \in \Upsilon$ est faisable avec un assignement de priorité *PF*, alors il est toujours faisable avec l'assignement de priorité *RM*. Plus précisément si τ est *PF*-faisable, alors le scénario d'activation synchrone du Lemme précédent conduit à un ordonnancement valide avec *PF*. Soit deux tâches τ_i et τ_{i+1} de τ ayant des priorités adjacentes mais ne respectant pas l'ordre *RM* (par exemple τ_i de priorité supérieure à τ_{i+1} => $T_i > T_{i+1}$). [Liu and Layland 1973] montrent qu'en permutant la priorité de ces deux tâches, ce scénario d'activation pire cas conduit toujours à un ordonnancement valide de τ. En effet ce Lemme précise que lorsque $T_i = D_i$, $\forall i \in [1, n]$, seule la première activation de chaque tache est critique, or :

- le temps de réponse des premières activations des tâches $\tau_{j \neq i, j \neq i+1}$ est inchangé,
- parmi les deux tâches permutées, τ_{i+1} ne peut qu'être avantagée par cette permutation. Par contre τ_i voit le temps de réponse de sa première activation augmenter, mais sans pouvoir dépasser T_{i+1} par l'hypothèse que τ est *PF*-faisable avant la permutation.

En réappliquant le raisonnement, l'assignement de priorité final est bien celui généré par *RM*.

IV.3.2.3. Faisabilité

En déroulant *RM* à partir du scénario synchrone et jusqu'à $max(T_i)$, [Leung and Whitehead 1982] notent que l'on obtient un test procédural pseudo-polynomial pour établir la faisabilité d'un trafic et calculer les pires temps de réponse des tâches (cf. Lemme 7 p. 35). Un autre résultat classique établit la condition suffisante, mais non nécessaire, polynomiale suivante basée sur la charge (cf. Définition 9 p. 15) et ne donnant pas les pires temps de réponse.

Théorème 6 - *[Liu and Layland 1973]. Soit un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN caractérisé par* $T_i = D_i, \forall i \in [1, n]$ *et* Π *contient RM.* $\forall (\tau \in \Upsilon)$, τ *est RM-faisable si :*

$$U \leq n(2^{1/n} - 1). \quad (12)$$

[Liu and Layland 1973] montrent que cette condition est suffisante en identifiant tout d'abord l'ensemble des trafics $\tau \in \Upsilon$ tels que l'augmentation de la durée d'exécution d'une tâche quelconque de τ conduit à un ordonnancement non valide par *RM* (on dit que τ est *RM*-critique, cf. Définition 12 p. 15). Par dérivation de l'expression de charge associée à ces trafics, ils établissent $n(2^{1/n} - 1)$, une borne sur la charge en dessous de laquelle tous les trafics sont faisables par *RM*. Notons que cette borne diminue avec le nombre de tâches du trafic. Elle vaut *1* si le trafic ne contient qu'une seule tâche et est proche de $log_e 2$ quand *n* tend vers l'infini. Liu et Layland remarquent de plus que cette borne vaut *1* si, $\forall i \in [1, n]$, $D_i = T_i$ et $(T_n / T_i) - \lfloor T_n / T_i \rfloor = 0$, c.à.d. lorsque T_n, la période de la tâche de plus faible priorité est multiple des autres périodes.

Notons que [Lehoczky 1990] étend ce résultat de faisabilité basé sur la charge au cas particulier où un rapport constant $\Delta > 0$ existe entre les périodes et les échéances relatives des tâches, c.à.d. $\forall i \in [1, n]$, $D_i = \Delta T_i$ (nous ne sommes plus alors dans le cas de Liu & Layland). Il montre alors qu'en moyenne une borne sur *U,* de l'ordre de *0.9,* conduit à la faisabilité du trafic.

Notons enfin que ces bornes de faisabilité basées sur la charge *U* ne sont valables que sous les hypothèses posées ($T_i = D_i$, $\forall i \in [1, n]$ pour [Liu and Layland 1973] et $D_i = \Delta T_i$, $\forall i \in [1, n]$ pour [Lehoczky 1990]) et n'ont pas pu être généralisées aux autres cas.

IV.3.3. Le cas $T_i \geq D_i$, $\forall i \in [1, n]$

IV.3.3.1. Scénario d'activation pires cas

Dans ce cas le Lemme 7 p. 35, est toujours valide mais en remplaçant $T_i = D_i$ par $T_i \geq D_i$ dans l'énoncé et t_0=*t*-T_i par t_0=*t*-D_i dans la preuve. Le scénario d'activation synchrone en *0* conduit alors toujours au scénario critique de chaque tâche sur sa première activation et à la faisabilité du trafic sur l'intervalle *[0,max(D_i)]*.

IV.3.3.2. Optimalité

Dans ce cas l'algorithme d'ordonnancement préemptif non-oisifs à priorités fixes *HPF/ Deadline-Monotonic* (noté ***DM*** dans la suite), qui assigne les priorités en ordre inversement proportionnel à la valeur de l'échéance relative des tâches est optimal. Ainsi la tâche ayant l'échéance relative la plus petite se voit attribuer la plus haute priorité (le tri est en $O(n\log_2 n)$).

Théorème 7 - *[Leung and Whitehead 1982]. DM est* **Σ-***optimal dans un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN caractérisé par* $T_i \geq D_i$, $\forall i \in [1, n]$ *et* Π *contient tout algorithme d'ordonnancement préemptif non-oisif à priorités fixes.*

La preuve montre que si un trafic non-concret $\tau \in \Upsilon$ est faisable avec une certaine assignation des priorités, alors il est toujours faisable avec *DM*. Le principe est le même qu'avec *RM* mais l'analyse est plus complexe puisque les échéances relatives peuvent ici être inférieures aux périodes. Plus précisément, [Leung and Whitehead 1982] partent d'une assignation de priorités quelconque pour laquelle τ est faisable. D'après le Lemme 7 p. 35, le scénario d'activation pire cas synchrone conduit encore à un ordonnancement valide et au pire temps de réponse de chaque tâche sur sa première activation. Soit deux tâches τ_i et τ_{i+1} ayant des priorités adjacentes mais ne respectant pas l'ordre *DM* (par exemple τ_i de priorité supérieure à τ_{i+1} => $D_i > D_{i+1}$). Si l'on inverse les priorités de τ_i et τ_{i+1} alors, les pires temps de réponse des tâches $\tau_{j \neq i, j \neq i+1}$ sont inchangés et la tâche τ_{i+1} devenant plus prioritaire, son pire temps de réponse ne peut que diminuer. Reste la tâche τ_i :

- avant l'inversion de priorités, nous avons par hypothèse: $x + C_i + C_{i+1} \leq D_{i+1}$, avec *x* la quantité de travail requise par les tâches $\tau_{1 \leq j < i}$ dans l'intervalle *[0, D_{i+1}]*.
- après l'inversion de priorités, il est suffisant (mais non nécessaire car on majore la quantité de travail requise et on minore l'échéance D_i) de garantir que:

$$\lfloor D_i / D_{i+1} \rfloor \cdot (x + C_{i+1}) + C_i \leq \lfloor D_i / D_{i+1} \rfloor \cdot D_{i+1}.$$

Comme la première équation implique la seconde, le trafic est toujours faisable. Si l'on réitère ce raisonnement tant que possible, l'assignement de priorité final est bien *DM*.

Notons qu'un trafic de ce type non faisable par *DM*, peut parfois être faisable en **non-préemptif** (cf. chapitre VIII.2.3. p. 82). L'optimalité de *DM* est donc relativement restreinte aussi bien sur l'ensemble Υ que sur Π. Ceci est impossible avec *EDF* dont la preuve d'optimalité couvre un référentiel d'ordonnancement bien plus général (cf. Théorème 1 p. 29).

IV.3.3.3. Faisabilité et pires temps de réponse

Contrairement au cas $T_i = D_i, \forall i \in [1, n]$ il n'existe pas ici de condition suffisante basée sur la charge U. La littérature focalise alors sur r_i, le pire temps de réponse de τ_i. Un test nécéssaire et suffisant évident est alors de tester : $\forall i \in [1, n], r_i \leq D_i$ (cf. Propriété 2 p. 11). Deux approches ont été proposées pour calculer ces pires temps de réponse :

- la procédure pseudo-polynomiale de [Leung and Whitehead 1982] est toujours valide ici en remplaçant *RM* par *DM* et en testant l'ordonnancement synchrone jusqu'à $\max(D_i)$.
- une approche plus formelle introduite par [Joseph and Pandya 1986] consiste à calculer r_i récursivement.

Théorème 8 - *[Joseph and Pandya 1986]. Soit un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN caractérisé par* $T_i \geq D_i, \forall i \in [1, n]$ *et* Π *contient PF, un algorithme d'ordonnancement préemptif non-oisif à priorités fixes.* $\forall (\tau \in \Upsilon)$, r_i, *le pire temps de réponse de toute tâche* τ_i *de* τ *par PF, est obtenue par l'équation récursive suivante (où hp(i) dénote l'ensemble des tâches de plus haute priorité que* τ_i*) :*

$$r_i^{(k+1)} = C_i + \sum_{j \in hp(i)} \left\lceil r_i^{(k)} / T_j \right\rceil C_j . \quad (13)$$

La preuve montre que l'éq. (13) est corrélée avec l'exécution de la première activation de τ_i dans le scénario d'activation synchrone. D'après le Lemme 7 p. 35, il n'est pas possible d'obtenir de plus mauvais temps de réponse en présence d'un autre scénario. La récursion s'arrête lorsque $r_i^{(k+1)} = r_i^{(k)} = r_i$ et peut être résolue par itération successives en partant de $r_i^{(1)} = C_i$. Il est facile de voir que la série $r_i^{(k)}$ est croissante et convergente avant D_i si τ_i est faisable (ou excède D_i dans le cas inverse).

IV.3.4. Le cas des trafics généraux

IV.3.4.1. Scénario d'activation pires cas

Pour ces trafics non-concrets τ, les périodes et les échéances relatives des tâches ne sont pas contraintes par des relations particulières (cf. Définition 2 p. 9). En conséquence, l'énoncé du Lemme 7 p. 35, est toujours valide mais la preuve doit être reformulée pour ces trafics puisque, T_i pouvant être inférieure à D_i, plusieurs activations de la tâche τ_i peuvent être en file d'attente simultanément sans pour autant rater d'échéance absolue. Autrement dit les pires temps de réponse ne sont pas nécessairement obtenus sur la première activation qui suit l'instant synchrone.

Ce problème est résolu par [Lehoczky 1990] grâce au concept de *level-i busy period*. Nous rappelons qu'une telle période est une période occupée durant laquelle seules des activations de tâches ayant une priorité supérieure ou égale à τ_i sont exécutées (cf. Définition 18 p. 19).

Lemme 8 - *[Lehoczky 1990]. Soit* τ*, un trafic non-concret et PF, un algorithme d'ordonnancement préemptif non-oisif à priorité fixe. Le pire temps de réponse de toute tâche* τ_i *de* τ *par PF se trouve dans la level-i busy period synchrone (c.à.d. résultante du scénario synchrone, cf. Definition 11 p. 15).*

La preuve de ce lemme montre que si un scénario d'activation ω conduit à rater l'échéance d'une activation de tâche τ_i en *t*, alors le scénario synchrone conduit aussi à rater une échéance. En effet soit *t'* le dernier instant précédent *t* pour lequel il n'y a pas d'activation de tâches ayant une priorité supérieure ou égale à τ_i en file d'attente. L'algorithme *PF* étant préemptif, non-oisif, il est donc continuellement occupé à exécuter des activations de tâches ayant une priorité supérieure ou égale à τ_i dans l'intervalle *[t', t]* (il s'agit donc d'une *level-i busy period*).

Si nous modifions le scénario initial ω en ramenant vers la gauche toutes les activations de tâches de priorités supérieures à τ_i de façon synchrones en *t'* et périodiques ensuite, alors τ_i rate toujours une échéance en *t* car le nombre d'activations de tâches ayant une priorité supérieure, et donc ordonnancées en priorité par *PF*, est maximisé dans l'intervalle *[t', t]*.

Si de plus, τ_i est aussi activée en *t'* et périodiquement ensuite, alors nous obtenons la *level-i busy period* résultant du scénario synchrone. Les instants d'exécution des activations de τ_i ne peuvent pas être modifiés dans *[t', t]* puisqu'il s'agit d'une *level-i busy period*. Par contre les échéances absolues de τ_i sont plus contraintes, τ_i rate donc toujours une échéance.

IV.3.4.2. Faisabilité et pires temps de réponse

Le résultat du Lemme précédent sur les *level-i busy period* conduit à l'établissement du test nécéssaire et suffisant suivant, basé sur une généralisation du calcul des pires temps de réponse introduit par le Théorème 8 p. 38.

Théorème 9 - *[Lehoczky 1990], [Burns et al. 1994]. Soit un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN et* Π *contient PF, un algorithme d'ordonnancement préemptif non-oisif à priorités fixes.* $\forall(\tau \in \Upsilon)$*, le pire temps de réponse* r_i *par PF de toute tâche* τ_i *de* τ *est obtenue par l'équation récursive suivante (où hp(i) dénote l'ensemble des tâches de plus haute priorité que* τ_i*) :*

$$r_i = max_{q = 1 \ldots Q_i}\{w_{i,q} - qT_i\}, \tag{14}$$

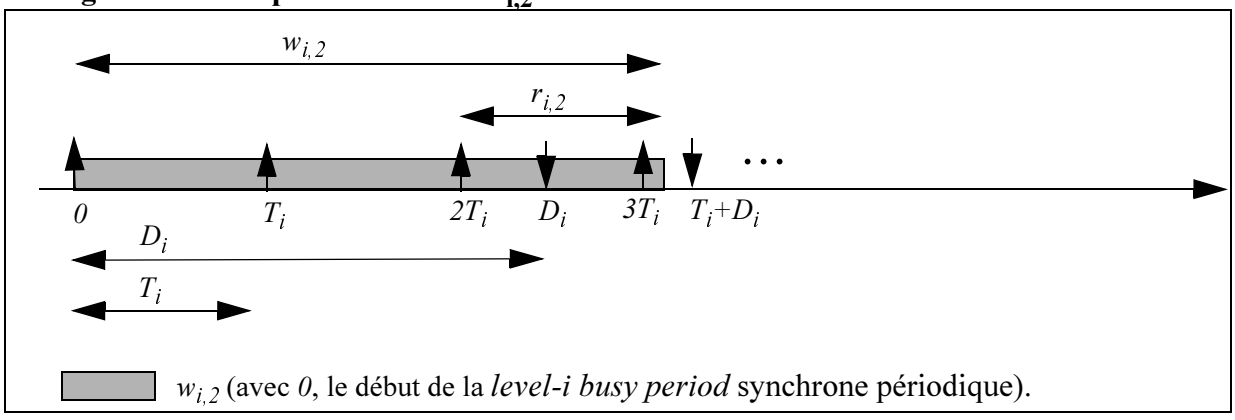
où Q_i *est la valeur minimale telle que* $w_{i,Q_i} \leq (Q_i + 1)T_i$ *et :*

$$w_{i,q}^{(k+1)} = (q+1)C_i + \sum_{j \in hp(i)} \left\lceil w_{i,q}^{(k)} / T_j \right\rceil C_j . \tag{15}$$

Partant du Lemme 8 p. 38, [Lehoczky 1990] et [Burns et al. 1994] montrent que l'énoncé de ce théorème est corrélé avec le scénario d'activation synchrone. Comme le pire temps de réponse de τ_i n'est pas nécessairement sur la première activation de τ_i en présence de trafics généraux τ, ils considèrent successivement toutes les fenêtres démarrant sur l'un des instants d'activation possible de τ_i dans sa *level-i busy period* synchrone. Si q est le numéro (partant de zéro) de l'activation de τ_i courante alors $w_{i,q}$ dénote la largeur de la période occupée contenant $q+1$ activation de τ_i, c.à.d. démarrant qT_i avant l'arrivée de la $(q+1)^{ième}$ activation de τ_i et se terminant sur sa fin d'exécution (cf. Figure 7 pour le cas où $q=2$). La valeur de $w_{i,q}$ est donnée par l'éq. (15) qui s'arrête quand $w_{i,q}^{(k+1)} = w_{i,q}^{(k)} = w_{i,q}$. Cette valeur donne bien l'instant de fin d'exécution de cette $(q+1)^{ième}$ activation, puisque celle ci peut être préemptée par une activation de tâche de priorité supérieure tant qu'elle n'a pas fini de s'exécuter.

Finalement, le pire temps de réponse de la tâche τ_i est donné par l'éq. (14) où Q_i est la valeur minimale de q telle que $Q_i T_i < w_{i,Q_i} \leq (Q_i + 1)T_i$. On trouve donc en-ligne $L_i = w_{i,Q_i}$ (cf. Definition 20 p. 20) et il reste à tester que $\forall i \in [1, n], r_i = max_{q = 1 \ldots Q_i}\{w_{i,q} - qT_i\} \leq D_i$.

Figure 7 : Exemple du calcul de $r_{i,2}$



Le coût de ce test pseudo-polynomial si $U \leq c < 1$ sera examiné lors du chapitre VI.2. p. 61.

IV.3.4.3. Optimalité

Concernant l'optimalité, [Lehoczky 1990] donne un contre-exemple prouvant que ni *RM* ni *DM* ne sont optimaux pour les trafics généraux τ. Une procédure coûteuse consiste bien sur à tester la faisabilité de τ par le Théorème 9 p. 39, sur toutes les permutations de priorités possibles. En pire cas, il y aurait alors *n!* (avec *n,* le nombre tâches non-concrètes dans τ) tests à effectuer avant de pouvoir conclure à la non faisabilité de τ. *Audsley* améliore cette procédure dans [Audsley 1991] en limitant à *2n* le nombre d'assignation de priorités à tester en pire cas[1].

Théorème 10 - *[Audsley 1991]. La procédure d'assignement de priorités, dite d'Audsley, est* Σ*-optimal dans un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN et* Π *contient tout algorithme d'ordonnancement préemptif, non-oisif à priorités fixes.*

```
----------/* assignation de priorités par Audsley */----------
```

$\tau = \{\tau_1, \ldots, \tau_n\}$;
for $j = n$ **downto** 1
  bool = TRUE;
  if $\exists\tau_k$ **de** τ **faisable par le Théorème 9 p. 39 à la priorité** j
      attribue à τ_k **la priorité** j; $\tau = \tau - \{\tau_k\}$; **bool = FALSE;**
  endif
  if bool= TRUE THEN
      EXIT; /* pas d'attribution de priorité faisable possible pour τ ***/**
  endif
endfor

La preuve d'*Audsley* commence par montrer que décroître la priorité d'une tâche τ_i ne peut augmenter les temps de réponse des autres tâches de τ. Intuitivement on voit bien que :

- toute tâche de τ de priorité intermédiaire entre l'ancienne et la nouvelle priorité de τ_i gagne un niveau de priorité. Son pire temps de réponse diminue donc par le Théorème 9 p. 39.
- toute tâche de τ de priorité supérieure à la nouvelle priorité (respectivement inférieure à l'ancienne priorité) de τ_i a un pire temps de réponse identique par le Théorème 9 p. 39.

La procédure commence donc par chercher à l'aide du Théorème 9 p. 39, une tâche de τ faisable avec la priorité la plus faible. Si plusieurs tâches sont faisables à ce niveau, il est possible de choisir la première trouvée (notée τ_k dans l'algorithme). En effet, la remplacer ultérieurement par l'une des autres tâches (notée τ_i) faisables à ce niveau laisse : (1) τ_i faisable par hypothèse ; (2) les autres tâches faisables car on vient de voir que décroître la priorité de τ_i ne peut augmenter les temps de réponse des autres tâches de τ.

La procédure réitère au niveau de priorité inférieure avec les tâches restantes. Elle continue ainsi jusqu'à ce qu'une priorité soit attribuée par tâche (le trafic τ est alors faisable) ou s'arrête à un certain niveau de priorité si aucune tâche n'est faisable à ce niveau (le trafic τ n'est pas faisable).

1. Cette procédure reste donc coûteuse et ne vérifie pas pleinement notre cahier des charges (cf. chapitre V.2.1. p. 47).

V. Extensions/discussions

L'état de l'art est important en préemptif. Nous introduisons juste ici le concept de *deadline busy period* pour *EDF*, symétrique avec celui de *level-i busy period* utilisée avec les priorités fixes, qui permet notamment de borner l'intervalle d'étude par tâche τ_i. Nous attirons ensuite rapidement l'attention du lecteur sur quelques limitations concernant les propriétés d'optimalité et de faisabilité avec les algorithmes à priorités fixes.

V.1. Earliest Deadline First

V.1.1. Deadline-d busy period

Le Théorème 9 p. 39, nous montre que les algorithmes d'ordonnancement préemptifs à priorités fixes font appel au concept de *level-i busy period* pour borner les intervalles d'étude lors de calculs de pires temps de réponse des tâches et de faisabilité des trafics. Rien de tel n'est proposé dans l'état de l'art avec *EDF* où l'on utilise directement *L*, la longueur de la période occupée synchrone du processeur (cf. chapitre III.2.1. p. 17). Le Lemme 2 p. 17, nous prouvant que *L* est la plus longue période occupée, nous focalisons maintenant sur le concept de *deadline-d busy period* pour montrer qu'il est possible d'en déduire $L_i \leq L$, la plus grande *deadline busy period* par tâche τ_i de τ. Les résultats proposés ici sont issus de [George et al. 1996][1].

V.1.1.1. Propriétés

Il est possible d'améliorer le Lemme 5 p. 30, grâce au concept de *deadline-d busy period* (cf. Définition 19 p. 19). Rappelons qu'une telle période est un intervalle où seules des activations de tâches ayant une échéance absolue inférieure ou égale à *d* sont exécutées.

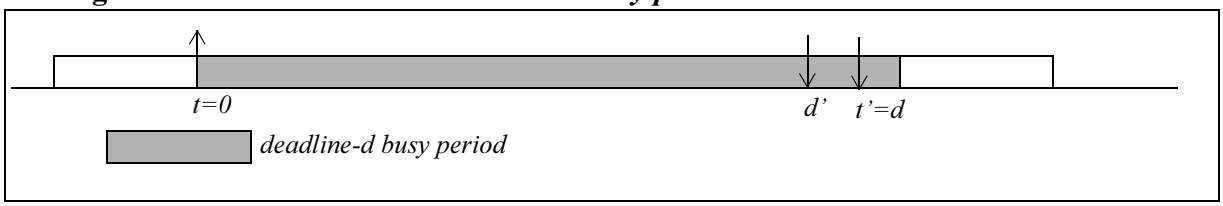
Lemme 9 - *[George et al. 1996]. Soit un trafic τ quelconque ordonnancé par EDF. Si une échéance absolue est ratée pour un certain scénario d'activation de τ alors $\exists d \geq 0$ tel que h(d)>d dans la deadline-d busy period résultant du scénario d'activation synchrone en 0 (cf. Definition 11 p. 15).*

Preuve. Soit ω, un scénario d'activation quelconque de τ conduisant une activation de la tâche τ_i à rater son échéance absolue en *t'* (cf. Figure 8). Soit t le dernier instant avant *t'* sans activation de tâche en file d'attente ayant une échéance absolue inférieure ou égale à *t'*. Par choix :

- t doit être l'instant d'activation d'une tâche (on pose *t*=0 et *t'=d* dans la suite),
- avec *EDF* il n'y a pas de période inoccupée du processeur entre 0 et d en non oisif et seules des activations de tâches ayant une échéance absolue inférieure ou égale à d sont exécutées dans l'intervalle $[0, d]$.

Ainsi 0 est le début de la *deadline-d busy period* résultant de la demande processeur dans l'intervalle $[0, d]$ et de longueur supérieure à d par l'hypothèse qu'une échéance est ratée en d.

Figure 8 : Faisabilité avec les *deadline busy periods*



Considérons maintenant le scénario d'activation pour lequel toutes les tâches, sauf τ_i, sont synchrones en 0 et activées périodiquement ensuite. La demande processeur n'a pu diminuer dans

1. et ont été établis avec notre co-auteur M. Spuri.

l'intervalle $[0, d]$ puisque le nombre d'activations de tâches ayant une échéance absolue dans cet intervalle n'a pu qu'augmenter. La longueur de la *deadline-d busy period* ne peut donc pas diminuer non plus et l'activation de τ_i considérée rate toujours son échéance absolue en d.

Si τ_i est aussi activée de façon synchrone en 0 et périodique ensuite, alors on a le scénario d'activation énoncé par ce lemme et la demande processeur est maximale durant $[0, d]$. Celle-ci vaut maintenant *h(d)* (cf. Définition 15 p. 16), qui donne la longueur maximale de la *deadline-d busy period* démarrant en 0. Soit d' la plus grande échéance absolue inférieure ou égale à d dans ce dernier scénario, on a $h(d)=h(d')$. Comme $h(d)$ est plus grand que $d \geq d'$, une échéance absolue est ratée au plus tard en d' dans cette *deadline-d busy period* démarrant en 0. ❒

Le pire scénario d'activation en terme de faisabilité par *EDF* est donc toujours le scénario synchrone, comme avec le Lemme 5 p. 30, mais il n'est plus nécessaire de tester toutes les échéances absolues dans l'intervalle $[0, L]$. Une première interprétation de ce résultat conduit en effet au test nécessaire et suffisant procédural suivant.

Théorème 11 - *Soit un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN et* Π *contient EDF. Pour tout* $\tau \in \Upsilon$*, on considère les activations de tâches (sur le scénario synchrone en 0) dont l'échéance absolue est inférieure à L et dont la fin d'exécution par EDF n'est pas précédée par l'exécution (même partielle) d'une activation ayant une échéance absolue supérieure.* τ *est faisable par EDF si et seulement si ces activations respectent leurs échéances..*

Preuve. Soit le scénario synchrone en 0 (cf. Définition 11 p. 15). Ce scénario est une instanciation concrète possible de τ, il est donc nécessaire de le tester. On note *d*, l'échéance absolue de l'activation de tâche en cours d'exécution par *EDF* à tout instant t.

Soit $t \geq 0$, tel que durant l'intervalle $[0, t]$ s'exécutent, même en partie, une ou des activations de tâches ayant une échéance absolue supérieure à *d*. Par définition la fin d'une telle exécution est le début d'une *deadline-d busy period* non synchrone.

Comme d'après le Lemme 9 p. 41, il est suffisant de tester les échéances absolues qui partent d'une *deadline busy period* synchrone périodique, il est donc inutile de tester *d*, l'échéance absolue de la tâche en cours d'exécution à l'instant *t*. Comme de plus *L* est la plus grande période occupée possible (cf. Lemme 2 p. 17), pour tout $t>L$, nous avons déjà testé toutes les échéances absolues partant d'une *deadline busy period* synchrone. ❒

Pour mettre en oeuvre cette procédure, il suffit de dérouler *EDF* sur le scénario d'activation synchrone en 0 et de tenir à jour *D*, la plus grande échéance absolue de toute activation ayant déjà été exécutée, même en partie. A tout instant on ne teste alors l'échéance de l'activation en cours d'exécution que lorsque celle-ci est supérieure ou égale à *D* et inférieure à *L*.

Une autre interprétation du Lemme 9 p. 41, est de calculer la valeur L_i^s de la plus longue *deadline busy period* **synchrone** possible par tâche τ_i de τ (cf. chapitre V.1.1.2. p. 43). Le Théorème 4 p. 32, basé sur la demande processeur, peut alors être borné par L_i^s par tâche τ_i.

Notons qu'il est aussi possible de calculer par tâche τ_i, la valeur L_i de la plus longue *deadline busy period* (**synchrone ou non-synchrone**) possible pour borner les intervalles d'étude nécessaires aux calculs des pires temps de réponse des tâches. Pour cela, vérifions tout d'abord que la période occupée mentionnée dans le Lemme 6 p. 33, est une *deadline-d busy period*.

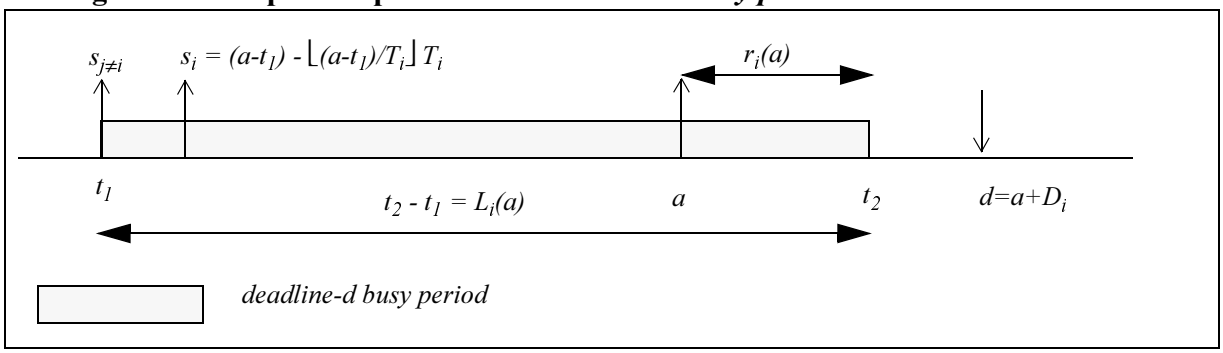
Lemme 10 - *[George et al. 1996]. Avec EDF, le pire temps de réponse d'une tâche* τ_i *de* τ *se trouve dans une deadline(a+*D_i*) busy period résultant d'une activation périodique des tâches, où a+*D_i *est l'échéance absolue d'une activation de* τ_i*, où toutes les autres tâches sont activées de façon synchrone en 0 et où* $L_i(a) > a$.

Preuve. Soit une activation de τ_i arrivée en a et d'échéance absolue $d = a + D_i$ (cf. Figure 9). Soit t_2, l'instant de fin d'exécution de cette activation par *EDF* et t_1, le dernier instant précédent t_2 tel qu'il n'y ait pas d'activation de tâche en file d'attente ayant une échéance absolue inférieure ou égale à d. Par choix nous avons $t_1 \leq a$ où t_1 est un instant d'activation de tâche.

Avec *EDF* il n'y a pas de période inoccupée du processeur durant $[t_1, t_2]$ en non oisif et cet intervalle est une période occupée durant laquelle seules des activations de tâches ayant une échéance absolue inférieure ou égale d sont exécutées, c.à.d. une *deadline-d busy period.*

Considérons maintenant le scénario d'activation périodique de l'Equation (9) p. 34 de $L_i(a)$, pour lequel les activations de τ_i arrivent en ..., a-$2T_i$, a-T_i, a, ..., et toutes les tâches, sauf τ_i, sont synchrones en t_1. Comme le nombre d'activations de tâches ayant une échéance absolue dans l'intervalle $[t_1, d]$ est maximisé par rapport au scénario initial, alors $L_i(a)$ donne la durée maximale de la *deadline-d busy period* et $L_i(a) > a$ pour pouvoir prétendre donner le pire temps de réponse de τ_i. En appliquant ce raisonnement quelque soit le scénario initial, on obtient nécessairement le pire temps de réponse de τ_i sur l'un d'entre eux. ❒

Figure 9 : Temps de réponse avec les *deadline busy periods*



V.1.1.2. Calcul de L_i

La Définition 20 p. 20, introduit la borne L_i. Avec *EDF*, le chapitre précédent nous indique :

- l'existence de L_i, la durée maximale d'une *deadline(a+D_i) busy period*, où les valeurs de a sont donc pertinentes pour le calcul du pire temps de réponse de τ_i (cf. Lemme 10 p. 42).
- que pour la faisabilité seule, il est possible de limiter cet durée à L_i^s résultant du scénario d'activation synchrone uniquement (cf. Lemme 9 p. 41).
- que $L_i^s \leq L_i \leq L$ car le calcul de L_i inclut le scénario d'activation synchrone périodique utilisé pour L_i^s et que L est la plus grande période occupée (cf. Lemme 2 p. 17).

En effet, une conséquence du Lemme 10 p. 42, est que L_i est la plus grande *deadline(a+D_i) busy period* vérifiant $L_i(a) > a$. Voyons maintenant comment accélerer la recherche de $L_i(a) = L_i$, c.à.d. du plus grand a vérifiant $L_i(a) > a$, à l'aide des deux propriétés suivantes.

Lemme 11 - *[George et al. 1996].* $D_i \leq D_j \Rightarrow L_i \leq L_j$.

Preuve. De l'Equation (9) p. 34, il existe $d = s_i + mT_i + D_i$ tel que $L_i = L_i(d - D_i)$. Pour tout $D_i \leq D_j$, nous avons alors :

$$\begin{aligned} L_i &= \sum_{D_k \leq d,\, k \neq i} min\left\{ \left\lceil \frac{L_i}{T_k} \right\rceil, 1 + \left\lfloor \frac{d - D_k}{T_k} \right\rfloor \right\} C_k + (m+1)C_i \\ &= \sum_{D_k \leq d,\, k \neq i,\, k \neq j} min\left\{ \left\lceil \frac{L_i}{T_k} \right\rceil, 1 + \left\lfloor \frac{d - D_k}{T_k} \right\rfloor \right\} C_k + (m+1)C_i \\ &\quad + min\left\{ \left\lceil \frac{L_i}{T_j} \right\rceil, 1 + \left\lfloor \frac{d - D_j}{T_j} \right\rfloor \right\} C_j. \end{aligned}$$

Comme $D_i \le D_j$, la dernière activation de τ_j ayant une échéance absolue inférieure ou égale à d a un instant d'activation inférieur ou égal à $d - D_i$ et donc à L_i. En admettant que $D_j \le d$, (dans le cas inverse, le résultat marche aussi trivialement), nous avons alors :

$$L_i = \sum_{D_k \le d,\, k \ne i,\, k \ne j} min\left\{\left\lceil \frac{L_i}{T_k} \right\rceil, 1 + \left\lfloor \frac{d - D_k}{T_k} \right\rfloor\right\} C_k + (m+1)C_i + \left(1 + \left\lfloor \frac{d - D_j}{T_j} \right\rfloor\right) C_j .$$

Considérons maintenant le scénario périodique pour lequel τ_i est synchrone c.à.d. $s_i = 0$, et $s_j = (d - D_j) - \lfloor (d - D_j)/T_j \rfloor T_j$, c.à.d. tel que τ_j a une activation ayant une échéance absolue en d, nous obtenons $L_j(d - D_j)$, l'un des scénarii à tester pour le calcul de L_j. Or :

$$L_i \le \sum_{D_k \le d,\, k \ne j} min\left\{\left\lceil \frac{L_i}{T_k} \right\rceil, 1 + \left\lfloor \frac{d - D_k}{T_k} \right\rfloor\right\} C_k + \left(1 + \left\lfloor \frac{d - D_j}{T_j} \right\rfloor\right) C_j = L_j(d - D_j) \le L_j.$$

En d'autres termes L_i, la quantité gauche de cette inéquation, est inférieure ou égale à la nouvelle *deadline-d busy period*, qui par définition est inférieure ou égale à L_j. ❐

Lemme 12 - *[George et al. 1996].* $\forall i, L_i(a)$ *est croissante en* a.

Preuve. Par définition, $L_i(a)$ est la plus petite racine de l'Equation (10) p. 34. Il vient que pour tout $t < L_i(a)$: $t < W_i(a, t) + (1 + \lfloor a/T_i \rfloor)C_i$. Soit $a' \ge a$ et $\forall t < L_i(a)$, nous avons :

$$t < W_i(a, t) + (1 + \lfloor a/T_i \rfloor)C_i \le W_i(a', t) + (1 + \lfloor a'/T_i \rfloor)C_i .$$

Comme $W_i(a, t)$ est croissante en a, il vient que $L_i(a') \ge L_i(a)$. ❐

On cherche L_i comme la valeur maximale de $L_i(a)$ vérifiant $L_i(a) > a$. On déduit pour cela des lemmes précédents une procédure récursive qui diminue la valeur de *a* tant que l'on ne vérifie pas $L_i(a) > a$. Soit un trafic non-concret τ trié dans l'ordre $D_i \le D_{i+1}$ pour toute tâche τ_i de τ et *E*, l'ensemble des échéance absolues du scénario d'activation synchrone :

$$E = \bigcup_{i=1}^{n} \{mTi + D_i, m \ge 0\} = \{e_1, e_2, \ldots\}$$

Pour calculer un L_i particulier, démarrons avec un scénario périodique où toutes les tâches, sauf τ_i, sont activées de façon synchrones en *0* (cf. Lemme 10 p. 42). "Calons" l'échéance absolue d'une des activations de τ_i sur la plus grande valeur $e_k \le L_{i+1} - C_i + D_i$. Au delà, d'après le Lemme 11 p. 43, il n'est pas possible d'obtenir L_i. Si $L_i(e_k - D_i) > e_k - D_i$, alors la *deadline-*$e_k$ *busy period* inclu l'activation de τ_i arrivée en $e_k - D_i$, nous avons donc trouvé L_i. Dans le cas contraire, comme d'après le Lemme 12 p. 44, il ne peut y avoir de plus grande *deadline busy period* pour τ_i, le calcul peut être repris en boucle en calant à chaque fois l'échéance absolue d'une activation de τ_i sur la plus grande valeur $e_k' \le L_i(e_k - D_i) - C_i + D_i$.

---------- Calcul des L_i [George et al. 1996] ----------

$L_{n+1} = L$;
for i=n **down to** 1
soit k tel que $e_k \le L_{i+1} - C_i + D_i < e_{k+1}$;
a = $e_k - D_i$;
while $L_i(a) \le a$
    soit k tel que $e_k \le L_i(a) - C_i + D_i < e_{k+1}$;
    a = $e_k - D_i$;
end while
$L_i = L_i(a)$;
end for

Pour la faisabilité seule la valeur de L_i calculée par l'algorithme précédent se limite au calcul de $L_i^s \leq L_i$ (où toutes les tâches sont synchrones et périodiques). Malheureusement, en se limitant ainsi, nous perdons la propriété du Lemme 11 p. 43, puisque l'on ne peut plus modifier le scénario d'activation initial. Pour toute tâche τ_i, la valeur de L_i^s peut cependant être calculée en initialisant systématiquement le calcul sur le majorant L de L_i^s (et non plus sur L_{i+1}^s):

---------- Calcul de L_i^s [George et al. 1996] ----------

a = $\lceil (L - D_i)/T_i \rceil T_i$;
while $L_i(a) \leq a$
    a = $\lceil (L_i(a) - D_i)/T_i \rceil T_i$;
end while
$L_i^s = L_i(a)$;

Ce dernier calcul de L_i^s peut sembler coûteux. Nous avons en effet vu qu'il était possible d'établir la faisabilité sur des intervalles certes plus grands mais sans sans passer par des calculs récursifs (cf. Théorème 4 p. 32). Par contre, le calcul préalable des L_i semble intéressant pour le calcul des pires temps de réponse qui font explicitement usage de formules récursives (cf. chapitre IV.2.4. p. 33). Des questions ouvertes subsistent :

- est-il possible de trouver des procédures de calcul préalable des L_i plus rapides et/ou en-ligne comme avec les priorités fixes (cf. chapitre IV.3.4.2. p. 39) ?
- existe-t-il une formule de $L_i(a)$ qui donne une valeur exacte de la fin d'exécution de l'activation de tâche arrivée en a pour toutes les valeurs de a. Rapppelons que lors du chapitre IV.2.4. p. 33, nous avons détecté des valeurs aberrantes, qui ne peuvent cependant remettre en cause le calcul des pires temps de réponse.

V.1.2. Optimalité et faisabilité

Avec *EDF*, nous savons de l'état de l'art que contrairement aux priorités fixes, il n'est pas nécessaire de passer par un calcul des pires temps de réponse des tâches pour calculer la faisabilité d'un trafic. Il est possible de tester le prédicat *h(t)*, égal à la demande processeur résultant du scénario synchrone, sur un intervalle :

$$h(t) = \sum_{i=1}^{n} max(0, 1 + \lfloor (t - D_i)/T_i \rfloor) C_i \qquad \text{(cf. Définition 15 p. 16).} \qquad (16)$$

Nous avons vu deux approches pour borner l'intervalle d'étude (manipulations algébriques de la demande processeur et périodes occupées). Ajoutons y une reformulation originale de la borne sur *L*, pour les valeurs de *t* pertinentes, proposée par J.-F. Hermant dans [Hermant et al. 1996].

Lemme 13 - *[Hermant et al. 1996], [Hermant 1998].*
$\forall \boldsymbol{t}, \boldsymbol{0} \leq \boldsymbol{t} \leq L, \quad \Phi(\boldsymbol{t}) \geq \boldsymbol{0} \quad \Rightarrow \forall \boldsymbol{t}, \boldsymbol{t} \geq \boldsymbol{0}, \quad \Phi(\boldsymbol{t}) \geq \boldsymbol{0} \quad$ *avec* $\Phi(t) = t - h(t)$.

La preuve montre que si $\Delta_L(t) = \Phi(t + L) - \Phi(t)$, alors :

$$\Delta_L(t) = L - \sum_{j=1}^{n} \left(max\left\{0, 1 + \left\lfloor \frac{t + L - D_j}{T_j} \right\rfloor\right\} - max\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} \right) C_j$$

$$\geq L - \sum_{j=1}^{n} max\left\{0, \left\lfloor \frac{L}{T_j} + \frac{t - D_j}{T_j} - \left\lfloor \frac{t - D_j}{T_j} \right\rfloor \right\rfloor\right\} C_j$$

$$\geq L - \sum_{j=1}^{n} \left\lceil \frac{L}{T_j} \right\rceil C_j$$

Comme L est la première racine de l'équation $L = \sum_{j=1}^{n} \lceil L/T_j \rceil C_j$ (cf. chapitre III.2.1. p. 17) alors $\Delta_L(t) = \Phi(t+L) - \Phi(t) \geq 0$. Si $\forall t, 0 \leq t \leq L, \Phi(t) \geq 0$, le trafic est donc faisable par *EDF*.

En intégrant les bornes précédentes à état de l'art, nous obtenons un résultat d'optimalité et un test nécéssaire et suffisant pour la faisabilité de tout trafic non-concret τ par *EDF*.

Théorème 12 - *[Shin and Zheng 1994], [George et al. 1996], [Hermant et al. 1996]. Soit un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN et* Π *contient tout algorithme d'ordonnancement. EDF est* Σ*-optimal et* $\forall(\tau \in \Upsilon)$, τ *est EDF-faisable si et seulement si*

$$\forall t \in S, \quad h(t) \leq t \quad \text{où } h(t) = \sum_{i=1}^{n} max(0, 1 + \lfloor (t - D_i)/T_i \rfloor) C_i,$$

$$S = \left(\bigcup_{i=1}^{n} \left\{ kT_i + D_i, 0 \leq k \leq \lceil (min\{\mu, L, L_i^s\} - D_i)/T_i \rceil + 1 \right\} \right),$$

$$\mu = max\left\{ max_{1 \leq j \leq n}\{D_j\}, \sum_{j=1}^{n} (T_j - D_j)U/(1-U) \right\},$$

$$L = \sum_{j=1}^{n} \lceil L/T_j \rceil C_j \quad \textit{(la première racine de cette équation),}$$

et L_i^s*, la plus grande deadline busy period synchrone pour la tâche* τ_i.

Preuve. $h(t)$ mesure la demande processeur maximale dans l'intervalle $[0, t]$ à partir du scénario d'activation synchrone en 0 (cf. Définition 15 p. 16). Cette condition est nécessaire pour tout algorithme (cf. Lemme 1 p. 16) et est suffisante pour *EDF*, qui est donc Σ-optimal, car :

- une conséquence du Lemme 9 p. 41, est que si un scénario d'activation rate une échéance avec *EDF*, alors il existe un instant *t* sur le scénario synchrone en 0 conduisant à vérifier que *h(t)>t*.
- $h(t)$ étant une fonction discontinue, $h(t)/t$ décroît entre un point de discontinuité et le point suivant. Seuls les points de discontinuité dans l'ensemble *S* sont donc pertinents.
- une deuxième conséquence du Lemme 9 p. 41, est la borne L_i^s sur l'intervalle d'étude (cf. chapitre V.1.1.2. p. 43). La borne L vient du Lemme 5 p. 30, ou du Lemme 13 p. 45. Notons que $L_i^s \leq L$, mais le calcul proposé pour L_i^s étant procédural, nous conservons la borne *L*.
- La borne μ vient du Théorème 4 p. 32. Plus précisément, si le test est vérifié dans l'intervalle *[0,*μ*]* mais rate une échéance à un instant $t>\mu$, alors comme par hypothèse $\mu \geq Max_{1 \leq j \leq n}\{D_j\}$, on obtient une contradiction puisque :

$$t < h(t) \leq \sum_{i=1}^{n} \left(\frac{t + T_i - D_i}{T_i} \right) C_i \leq tU + \sum_{i=1}^{n} \left(1 - \frac{D_i}{T_i} \right) C_i \quad \text{c.à.d. } t < \frac{\sum_{i=1}^{n} (T_j - D_j)U}{1-U}. \square$$

Ce test est pseudo-polynomial si $U \leq c < 1$ (cf. chapitre VI.2. p. 61) et confirme l'optimalité de *EDF* (cf. Théorème 1 p. 29) en présence des trafics non-concrets qui nous intéressent.

V.1.3. Faisabilité basée sur le calcul des pires temps de réponse

Cette approche garantit la faisabilité d'un trafic par *EDF* en vérifiant que $\forall i \in [1, n], r_i \leq D_i$. L'état de l'art (cf. chapitre IV.2.4. p. 33) et le Lemme 10 p. 42 nous indiquent que r_i résulte d'une *deadline(a+D$_i$) busy period* où toutes les tâches sont activées périodiquement et sont, sauf τ_i, synchrones en *0* alors que τ_i est activée en ..., *a-2T$_i$*, *a-T$_i$*, *a*, *a+T$_i$*... . La longueur de *L$_i$(a)* est alors donnée par l'Equation (9) p. 34 :

$$L_i(a) = (1 + \lfloor a/T_i \rfloor) C_i + \sum_{j \neq i} min\{ \lceil L_i(a)/T_j \rceil, max\{0, 1 + \lfloor (a + D_i - D_j)/T_j \rfloor\}\} C_j \quad (17)$$

Nous avons vu comment borner les valeurs de *a* par L_i. De même que pour la faisabilité, ajoutons y la reformulation de la borne *L* proposée par J. F. Hermant dans [Hermant et al. 1996].

Lemme 14 - *[Hermant et al. 1996], [Hermant 1998].* $\forall a\ ,\ a \geq 0\ ,\quad r_i(a+L) \leq r_i(a)$

La preuve montre que si par hypothèse $L_i^{(k)}(a+L) - L_i^{(k)}(a) \leq L$, alors :

$$L_i^{(k+1)}(a+L) = (1+\lfloor (a+L)/T_i \rfloor)C_i + \sum_{j \neq i} min\left\{\left\lceil L_i^{(k)}(a+L)/T_j \right\rceil, max\{0, 1+\lfloor (a+L+D_i-D_j)/T_j \rfloor\}\right\}C_j$$

$$\leq (1+\lfloor a/T_i \rfloor)C_i + \sum_{j \neq i} min\left\{\left\lceil L_i^{(k)}(a)/T_j \right\rceil, max\{0, 1+\lfloor (a+D_i-D_j)/T_j \rfloor\}\right\}C_j + \sum_j \lceil L/T_j \rceil C_j$$

$$= L_i^{(k+1)}(a) + L\ .$$

Comme par définition $L_i^{(0)}(a+L) - L_i^{(0)}(a) \leq L$, alors par récurrence $L_i(a+L) - L_i(a) \leq L$.

Comme de plus $r_i(a+L) - r_i(a) = L_i(a+L) - L_i(a) - L$, le résultat est prouvé.

En intégrant les bornes précédentes à l'état de l'art, nous obtenons le test nécéssaire et suffisant suivant basé sur le calcul des pires temps de réponse des tâches d'un trafic non-concret τ.

Théorème 13 - *[Spuri 1996], [George et al. 1996], [Hermant et al. 1996]. Soit un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN et* Π *contient EDF.* $\forall(\tau \in \Upsilon)$, τ *est EDF-faisable si et seulement si:*

$$\forall i \in [1,n],\ r_i \leq D_i \text{ où } \quad r_i = max_{a \in S}\{L_i(a) - a\} \quad \textit{avec:}$$

$$L_i(a) = (1+\lfloor a/T_i \rfloor)C_i + \sum_{j \neq i} min\{\lceil (L_i(a))/T_j \rceil, max\{0, 1+\lfloor (a+D_i-D_j)/T_j \rfloor\}\}C_j\ ,$$

$$S = (\bigcup_{j=1}^{n}\{kT_j + D_j - D_i,\ 0 \leq k \leq \lfloor min\{L, L_i\}/T_j \rfloor\})\ ,$$

$$L = \sum_{j=1}^{n} \lceil L/T_j \rceil C_j \quad \textit{(la première racine de cette équation),}$$

L_i *(la plus grande deadline busy period pour la tâche* τ_i *,* $\forall i \in [1,n]$*).*

Preuve. $L_i(a)$ est la durée d'une *deadline(a+D_i) busy period* pour l'un des scénarii d'activation du Lemme 10 p. 42. Cette condition est trivialement nécessaire pour *EDF* car ces scénarii sont des instanciations concrètes possibles de tout $\tau \in \Upsilon$ et $\forall i \in [1,n]$, $\forall a$, $r_i(a) = L_i(a) - a \leq D_i$ si τ est faisable par *EDF*. Elle est aussi suffisante pour *EDF* car tout temps de réponse dans un scénario d'activation quelconque ne peut qu'augmenter en le ramenant au scénario d'activation correspondant du Lemme 10 p. 42. Il vaut alors $L_i(a) - a$.

Une deuxième conséquence de ce lemme est la borne L_i sur l'intervalle d'étude (cf. chapitre V.1.1.2. p. 43). La borne L vient du Lemme 14 p. 47. Notons que $L_i \leq L$ mais le calcul proposé pour L_i étant procédural, nous conservons la borne L. ❐

Ce test est pseudo-polynomial si $U \leq c < 1$ (cf. chapitre VI.2. p. 61).

V.2. Priorités fixes

Des propriétés d'optimalité et de faisabilité sont connus pour les priorités fixes (cf. chapitre IV.3. p. 35). Nous attirons juste ici l'attention du lecteur sur quelques limitations.

V.2.1. Une optimalité limitée

Les résultats d'optimalité pour les priorités fixes n'ont pas la généralité du résultat sur *EDF*. Illustrons ceci en présence de trafic concrets ω puis de trafics non-concrets τ.

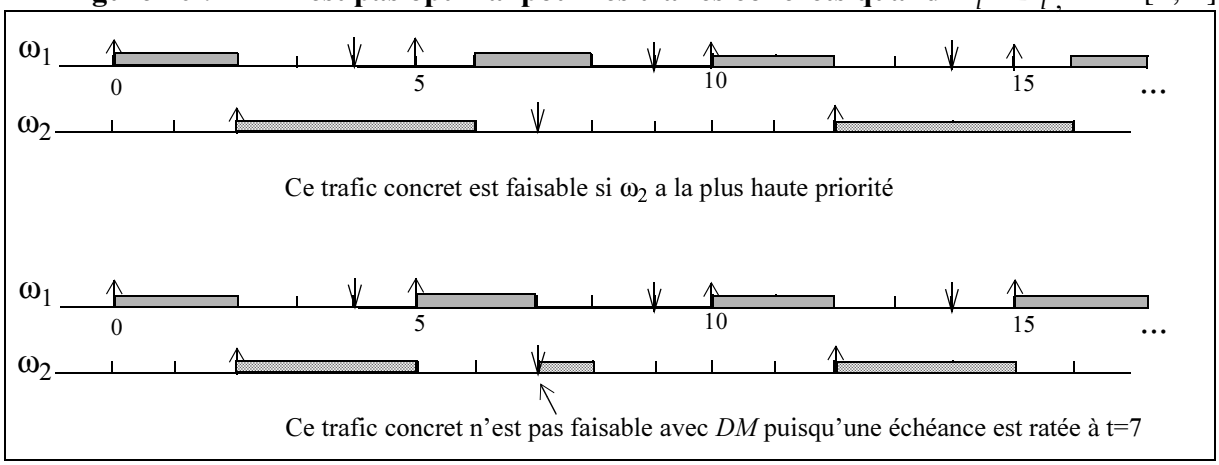
V.2.1.1. Trafic concrets

Si le référentiel d'ordonnancement Σ contient *EDF*, nous savons que cet algorithme reste Σ-optimal en présence de trafics concrets ω (c.à.d. de tout scénario d'activation) car la preuve du Théorème 1 p. 29, montre que tout ordonnancement valide peut être ramené par permutation à un ordonnancement *EDF* valide. Cette optimalité s'applique donc avec une définition de l'optimalité très forte (cf. Définition 29 p. 23).

Il est trivialement possible par contre de montrer que cela est faux pour les résultats connus sur les priorités fixes. Ainsi, contrairement au cas de trafics non-concrets Théorème 7 p. 37, *DM* n'est pas optimal parmi les algorithmes à priorités fixes préemptifs dans le cas $T_i \geq D_i$, $\forall i \in [1, n]$ pour les trafics concrets. Soit le scénario d'activation ω suivant (cf Figure 10) :

$$\omega_1(C_1=2, T_1=5, D_1=4, s_1=0),\ \omega_2(C_2=4, T_2=10, D_2=5, s_2=2).$$

Figure 10 : *DM* n'est pas optimal pour les trafics concrets quand $T_i \geq D_i$, $\forall i \in [1, n]$



V.2.1.2. Trafics non-concrets

La procédure d'*Audsley*, optimale parmi les algorithmes à priorités fixes en présence de trafics généraux τ (c.à.d. non-concrets et ne présentant pas de relations entre les périodes et les échéances des tâches), multiplie cependant le coût d'établissement de la faisabilité d'un trafic non-concret par $O(n^2)$ (cf. chapitre IV.3.4.3. p. 40). Elle correspond en fait à une amélioration d'une procédure optimale en *n!* (avec *n* le nombre de tâches), qui testerait toutes les permutations possibles pour assigner les priorités.

La question de savoir s'il existe un algorithme à priorités fixes qui assigne les priorités de façon optimale et à faible coût (comme *RM* ou *DM*) en présence de trafics généraux reste ouverte. Une question plus limitée est de savoir s'il peut exister une relation de dominance où :

Définition 35 - *Soit* $\Sigma=(\Upsilon,\Pi)$ *un référentiel d'ordonnancement. Soit* $(P, Q) \in \Pi^2$*, P est dit* Υ***-dominant(Q)*** *si et seulement si:* $(\forall \tau \in \Upsilon), (Q(\tau) \Rightarrow P(\tau))$*, où* τ *est un trafic non-concret et* $Q(\tau)$ *(respectivement* $P(\tau)$*) signifie que* τ *est Q-faisable (respectivement P-faisable).*

En d'autres termes, cette propriété énonce que tous les trafics ordonnançables par l'algorithme *Q* le sont aussi par *P*. En particulier, un algorithme Σ-optimal (cf. Definition 24 p. 21) est ainsi Υ-dominant sur tous les algorithmes du référentiel Σ.

L'idée qui vient immédiatement est de vérifier s'il n'existe pas une propriété de dominance entre *DM* et *RM* dans le cas général. En effet nous savons déjà que *DM* est Σ-optimal lorsque Υ contient tout TPN caractérisé par $T_i \geq D_i,\ \forall i \in [1, n]$ et Π contient tout algorithme préemptif non-oisif à priorités fixes (cf. Théorème 7 p. 37). Nous savons donc que *DM* est Υ-dominant(*RM*) dans ce cas. Malheureusement, ceci n'est plus vrai dans le cas général où il n'est pas possible de montrer une telle propriété entre *DM* et *RM*, dans un sens ou dans l'autre. Soit le trafic non-concret τ suivant :

$\tau_1(C_1=1, T_1=8, D_1=24)$, $\tau_2(C_2=4, T_2=12, D_2=22)$, $\tau_3(C_3=8, T_3=16, D_3=20)$.

En utilisant le Théorème 14 p. 50 et l'assignation de priorités *RM*, nous obtenons par ordre de priorités : r_1=1 ; r_2=5 ; r_3=19. τ est donc faisable par *RM*. En utilisant l'assignation de priorités *DM*, nous obtenons par ordre de priorités : r_3=8 ; r_2=12 ; r_1=29>24. τ n'est donc pas faisable par *DM*.

Dans le même temps, si nous modifions juste les durées d'exécution des tâches de τ de façon à obtenir :

$\tau_1(C_1=2, T_1=8, D_1=24)$, $\tau_2(C_2=6, T_2=12, D_2=22)$, $\tau_3(C_3=3, T_3=16, D_3=20)$.

En utilisant le Théorème 14 p. 50 et l'assignation de priorités *RM*, nous obtenons par ordre de priorités : r_1=2 ; r_2=8 ; r_3=21>20. τ n'est donc pas faisable par *RM*. En utilisant l'assignation de priorités *DM*, nous obtenons par ordre de priorités : r_3=3 ; r_2=9 ; r_1=15. τ est donc faisable par *DM*.

Le premier cas nous dit que *DM* n'est pas Υ-dominant(*RM*), le deuxième que *RM* n'est pas Υ-dominant(*DM*). Ces trafics vérifient $T_i < D_i,\ \forall i \in [1, n]$, cette conclusion est donc à fortiori vraie dans le cas général où les périodes sont indépendantes des échéances. Notons :

- que l'on obtient les mêmes conclusions en ajoutant la tâche $\tau_4(C_4=1, T_4=48, D_4=47)$. Les temps de réponse des autres tâches sont identiques et r_4 vaut 47 dans le premier cas et 46 dans le deuxième. Cette tâche vérifiant $T_4 > D_4$, cela confirme bien le résultat dans le cas général.
- que ceci n'interdit pas l'existence d'un algorithme qui assignerait les priorités fixes de façon optimale et à faible coût en présence de trafics généraux. Nous n'en avons pas trouvé et nous contenterons de conseiller *DM* (plutôt que *RM*) dans le cas général pour les raisons suivantes :
  - *DM* est Υ-dominant(*RM*) dans le cas $T_i \geq D_i,\ \forall i \in [1, n]$.
  - il semble relativement plus difficile de trouver des exemples en faveur de *RM*, plutôt que *DM*, dans le cas général (nous sommes conscient des limites d'un tel argument).

V.2.2. Faisabilité uniquement par un calcul de pires temps de réponse

L'état de l'art n'offre aucune piste avec les priorités fixes pour établir la faisabilité d'un trafic sans passer explicitement, ou implicitement, par un calcul de pires temps de réponse. Sans interdire un tel résultat, il est montré dans [Hermant et al. 1996] que, contrairement à *EDF* (cf. Théorème 12 p. 46), il est impossible d'énoncer un test nécéssaire et suffisant pour les priorités fixes sous la forme $((\forall \tau \in \Upsilon) \wedge (\forall t \in S(\tau)))(f(t, \tau) \leq t)$ (ou f est une fonction positive convexe pour τ et $S(\tau)$ est un ensemble de réels). Ce résultat original, développé dans [Leboucher 1998], se rapporte aux notions de critère d'optimalité permettant d'identifier une Σ-région convexe.

L'état de l'art garantit par contre la faisabilité d'un trafic en vérifiant que $\forall i \in [1, n]$, $r_i \leq D_i$ où $r_i = max\{w_{i,q} - qT_i\}$ résulte d'une *level-i busy period* synchrone (cf. Théorème 9 p. 39). La longueur de $w_{i,q}$ étant donnée par l'Equation (15) p. 39 :

$$w_{i,q} = (q+1)C_i + \sum_{j \in hp(i)} \lceil w_{i,q}/T_j \rceil C_j \qquad (18)$$

Le calcul de L_i, la plus grande *level-i busy period* permettant de borner l'intervalle d'étude, adapte le calcul de L du chapitre III.2.1. p. 17, en ne considérant que les instants d'activations des tâches de priorité supérieure ou égale à τ_i . Une reformulation originale de cette borne, pour les valeurs de q pertinentes, est proposée par J. F. Hermant dans [Hermant et al. 1996].

Lemme 15 - *[Hermant et al. 1996], [Hermant 1998].*
$\forall q \in \mathbb{N}$, $r_{i,\,q+Q_i+1} \le r_{i,\,q}$ *avec* $Q_i = \lceil L_i/T_i \rceil - 1$ *et* $L_i = \sum_{j \in hp(i) \cup \{i\}} \lceil L_i/T_j \rceil C_j$.
$Q_i + 1 = \lceil L_i/T_i \rceil <=> Q_i T_i < L_i \le (Q_i + 1)T_i$ avec $w_{i,\,Q_i} = L_i$. La preuve montre alors que :

$$w_{i,\,q} + w_{i,\,Q_i} = (q + Q_i + 2)C_i + \sum_{j \in hp(i)} (\lceil w_{i,\,q}/T_j \rceil + \lceil w_{i,\,Q_i}/T_j \rceil) C_j$$

$$\ge (q + Q_i + 2)C_i + \sum_{j \in hp(i)} \lceil (w_{i,\,q} + w_{i,\,Q_i})/T_j \rceil C_j$$

Comme $w_{i,\,q+Q_i+1} = (q + Q_i + 2)C_i + \sum_{j \in hp(i)} \lceil w_{i,\,q+Q_i+1}/T_j \rceil C_j$,
$w_{i,\,q} + w_{i,\,Q_i} < w_{i,\,q+Q_i+1}$ est impossible.

Nous avons donc $w_{i,\,q+Q_i+1} - w_{i,\,q} \le L_i$ et le résultat est prouvé puisque :

$$r_{i,\,q+Q_i+1} - r_{i,\,q} = w_{i,\,q+Q_i+1} - w_{i,\,q} - (Q_i + 1)T_i \le L_i - \lceil L_i/T_i \rceil T_i \le 0 .$$

En intégrant cette borne à l'état de l'art et en testant les échéances, la condition nécessaire et suffisante permettant d'établir la faisabilité d'un trafic non-concret par un algorithme d'ordonnancement à priorités fixes s'énonce comme suit.

Théorème 14 - *[Lehoczky 1990], [Burns et al. 1994], [Hermant et al. 1996]. Soit un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN et* Π *contient P, un algorithme d'ordonnancement préemptif non-oisif à priorités fixes.* $\forall (\tau \in \Upsilon)$, τ *est P-faisable si et seulement si:*

$$\forall i \in [1, n],\ r_i \le D_i \quad \text{où } r_i = max_{0 \le q \le Q_i} \{w_{i,\,q} - qT_i\},$$

$$w_{i,\,q} = (q + 1)C_i + \sum_{j \in hp(i)} \lceil w_{i,\,q}/T_j \rceil C_j ,$$

$Q_i = \lceil L_i/T_i \rceil - 1$, *et* $L_i = \sum_{j \in hp(i) \cup \{i\}} \lceil L_i/T_j \rceil C_j$ *(la première racine de cette équation).*

Preuve. Idem Théorème 9 p. 39, avec le Lemme 15 pour les valeurs de $0 \le q \le Q_i$ pertinentes. ❐

Ce test est pseudo-polynomial si $U \le c < 1$ (cf. chapitre VI.2. p. 61).

VI. Eléments de comparaison de performances des algorithmes

Depuis [Liu and Layland 1973], la littérature en ordonnancement temps réel s'est largement spécialisée par type d'algorithme, typiquement *EDF* contre *RM*, ou par enrichissements des trafics (contraintes temporelles, relations de précédences, ressources partagées...). Peu d'éléments de comparaison des performances existent cependant. Aussi, sans modifier notre modèle de trafic non-concret τ simple (cf. Définition 2 p. 9), examinerons-nous ici l'efficacité des algorithmes et le coût des tests associés. Quelques applications numériques seront proposées au chapitre X. p. 93.

VI.1. Efficacité des algorithmes

En établissant notre cahier des charges, nous nous sommes proposé d'assouplir la propriété d'optimalité. $\varepsilon(P)$, l'efficacité d'un algorithme *P*, en particulier donne une mesure de la proximité à l'optimalité de *P* (cf. Propriété 3 p. 11). Après un rapide état de l'art, la base théorique pour cette mesure, introduite par L. Leboucher dans [Hermant et al. 1996][1], est rappelée. *EDF* étant l'algorithme le plus efficace, nous spécialiserons la minoration et l'interprétation de l'efficacité des algorithmes à priorités fixes proposée dans [Hermant et al. 1996] pour l'algorithme *DM*.

VI.1.1. Etat de l'art

En terme d'optimalité nous savons, par la propriété de dominance[2] et par restrictions successives du référentiel d'ordonnancement, que *EDF* (cf. Théorème 12 p. 46) domine *Audsley* (cf. Théorème 10 p. 40) qui domine *DM* (cf. Théorème 7 p. 37) qui domine[3] *RM* (cf. Théorème 5 p. 36). Par contre, à notre connaissance seuls quelques résultats basés sur U, la charge d'un trafic (cf. Définition 9 p. 15), peuvent être interprétés en terme de mesure d'efficacité dans la littérature.

En contexte préemptif centralisé, avec un modèle de trafics non-concrets τ contraints par $D_i = T_i$, $\forall i \in [1, n]$, [Liu and Layland 1973] montrent que $U \leq 1$ est un test de faisabilité nécéssaire et suffisant pour *EDF* (cf. Théorème 2 p. 30) et que $U \leq n(2^{1/n} - 1)$ est suffisant pour *RM* (cf. Théorème 6 p. 36). U peut donc être interprété, avec nos notations du chapitre III.3. p. 20, comme une mesure de l'efficacité dans un tel référentiel d'ordonnancement. *EDF* est alors Σ-optimal puisque $\varepsilon(EDF) = 1$. Par contre $n(2^{1/n} - 1)$ nous donne une minoration de $\varepsilon(RM)$, la proximité à $\varepsilon(EDF)$ par le meilleur algorithme à priorité fixe dans ce référentiel.

[Lehoczky 1990] étend ce résultat au cas $\forall i \in [1, n]$, $D_i = \Delta T_i$ (avec $\Delta > 0$). Sous cette hypothèse il établi une borne sur *U* (de l'ordre de *0.9* en moyenne) qui conduit à la faisabilité des trafics par *DM*. Plus généralement, lorsque l'on sort des hypothèses de [Liu and Layland 1973], nous ne connaissons pas de résultats permettant de borner l'efficacité des algorithmes. En particulier *U* ne semble pas toujours être un critère pertinent car il est trivial de trouver des trafics non faisables, même par *EDF*, qui présentent poutant une très faible charge. Par exemple si τ contient :

$\tau_1(C_1=2, T_1=1000, D_1=3)$ et $\tau_2(C_2=2, T_2=1000, D_2=3)$, on a U=0.004 et τ n'est pas faisable.

Notons que [Liu and Layland 1973] étudient aussi l'impact d'un trafic τ, ordonnancé par *RM* en premier plan, sur un autre trafic τ', ordonnancé par *EDF* quand τ ne réquisitionne pas la ressource processeur. La fonction *f(t)* définie comme le temps processeur disponible pour τ' dans l'intervalle *[0,t]* étant sous-linéaire[4], le Lemme 5 p. 30, reste donc valable si *0* est un instant synchrone (cf. Définition 11 p. 15). Le test résultant pour τ' étant non trivial, Liu & Layland montrent simplement par quelques exemples que la borne de faisabilité sur U est considérablement améliorée lorsque le nombre de tâches ordonnancées par *EDF*, plutôt que par *RM*, augmente.

1. Elle est développée dans [Leboucher 1998].
2. La Définition 35 p. 48, énonce qu'un algorithme en domine un autre quand il ordonnance au moins les mêmes trafics.
3. Attention, *DM* ne domine *RM* que dans le cas $T_i \geq D_i$, $\forall i \in [1, n]$ (cf. chapitre V.2.1. p. 47).
4. $f(T) \leq f(T + t) - f(t)$.

VI.1.2. Rappel théorique sur l'efficacité

Il semble donc que *EDF* soit un algorithme efficace. Afin de quantifier cette intuition en fonction du problème posé, rappelons la base théorique du calcul de l'efficacité des algorithmes.

VI.1.2.1. Structure algébrique

Le chapitre III.3. p. 20, introduit les concepts de référentiel d'ordonnancement $\Sigma = (\Upsilon, \Pi)$, de Σ-optimalité, de TPN et de couples *(T,D)* (cf. Définition 36 p. 52). Ceci nous a permis d'énoncer les résultats d'optimalité et de faisabilité sans ambiguïté. Cela permet aussi d'assimiler un référentiel d'ordonnancement préemptif à un espace vectoriel de dimension égale aux nombre de couple *(T,D)*. L'intérêt d'utiliser l'algèbre linéaire est de considérer l'efficacité comme une norme dans un tel espace. Tout d'abord, munissons nous d'une relation d'équivalence simple entre TPN.

Définition 36 - *[Hermant et al. 1996], [Leboucher 1998]. Deux TPN sont dit* ***équivalents en périodes et en échéances*** *si pour chacun des couples* (T,D) *représentés dans* Υ*, la somme des durées d'exécution des tâches caractérisées par le même couple* (T,D) *est égale pour les deux TPN.*

Cette relation d'équivalence n'est pas sans conséquence sur les algorithmes optimaux identifiés précédemment (*EDF*, *DM*...). Ainsi, l'ordonnancement du scénario synchrone (cf. Définition 11 p. 15), le plus dur en terme de faisabilité pour ces algorithmes, reste identique aprés passage au quotient[1] de l'ensemble Υ par cette relation.

Cette propriété est intéressante car le TPN "agrégé" obtenu par passage au quotient d'un trafic de Υ ne peut que réduire le nombre de tâches à traiter. D'autre part elle généralise le résultat, car si le TPN agrégé est faisable par un algorithme *P* compatible pour la faisabilité avec la propriété d'équivalence en période et en échéances, alors tout trafic équivalent en périodes et en échéances sera aussi faisable par *P*. Enfin L. Leboucher note que l'addition de TPN est ainsi rendue commutative et compatible avec la multiplication (c.à.d. $\tau+\tau=2\tau$, cf. Définition 26 p. 22).

Le TPN agrégé identifie donc une classe d'équivalence compatible pour la faisabilité avec un référentiel d'ordonnancement périodique $\Sigma = (\Upsilon, \Pi)$, dès lors que les ordonnancements générés par les algorithmes de Π ne sont pas modifiés aprés passage au quotient de Υ par cette relation.

Théorème 15 - *[Hermant et al. 1996], [Leboucher 1998]. Soit* $\Sigma = (\Upsilon, \Pi)$*, un référentiel d'ordonnancement périodique compatible pour la faisabilité avec la propriété d'équivalence en périodes et en échéances.* Υ*, un ETPN muni des opérateurs d'addition et de multiplication, est un espace vectoriel sur le corps des réels que l'on notera* $(\Upsilon, +, \cdot)$*.*

Σ est compatible avec la propriété d'équivalence en période et en échéances, Υ peut donc être rendu stable pour l'addition de TPN et pour la multiplication de TPN par des scalaires (cf. Définition 26 p. 22). Il est ainsi possible de donner une signification à la soustraction de TPN ou à des TPN ayant des durées d'exécution négatives.

La loi + est associative, commutative, admet des trafics symétriques et un élément neutre qui est le trafic $\tau_0 \in \Upsilon$ ayant une durée d'exécution nulle pour toutes ces tâches. La loi $\cdot$ se traduit par la multiplication d'un trafic par un scalaire ($\Upsilon \times \mathbb{R} \rightarrow \Upsilon$), c.à.d. des durées d'exécution de toutes les tâches du trafic par ce scalaire. Cette multiplication est distributive, associative et l'élément neutre est le scalaire *1*.

Concrètement, la dimension de l'espace est égale au nombre de couples (T,D) représentés dans Υ et tout $\tau \in \Upsilon$ est un vecteur de $(\Upsilon, +, \cdot)$. Soit la famille $\{e_{T,D}\}_{(T,D) \in T \times D}$ de vecteurs unitaires caractérisés chacun par l'un des couples (T,D) représentés dans Υ. Cette famille est libre et génératrice, elle est donc une base. Tout $\tau \in \Upsilon$ est alors combinaison linéaire des vecteurs de la base, ce qui peut s'écrire sous la forme $\tau = \sum_{(T,D) \in T \times D} C_{T,D} e_{T,D}$.

1. c.à.d. pour tout $\tau \in \Upsilon$, la sommation des durées d'exécution de toutes les tâches caractérisées par le même couple (T,D). Par abus de langage nous continuerons d'appeler Υ le passage au quotient de l'ensemble Υ.

Il est alors possible d'assimiler la mesure de l'efficacité des algorithmes d'ordonnancement dans Σ à un calcul de norme. Rappelons qu'une norme dans $(\Upsilon,+,\cdot)$ sera une application de Υ à valeur dans $\mathbb{R}^+$ (notée $f:\tau \rightarrow \|\tau\|$) qui possède les trois propriétés classiques suivantes. Pour tout $(\tau, \tau') \in \Upsilon^2$ et tout $\lambda \in \mathbb{R}$:

- $\|\tau\| = 0 \Leftrightarrow \tau = \tau_0$.
- $\|\lambda \cdot \tau\| = |\lambda| \cdot \|\tau\|$.
- $\|\tau + \tau'\| \leq \|\tau\| + \|\tau'\|$ (l'inégalité triangulaire).

Définition 37 - *[Hermant et al. 1996], [Leboucher 1998]. Soit $\Sigma = (\Upsilon, \Pi)$, un référentiel d'ordonnancement périodique compatible pour la faisabilité avec la propriété d'équivalence en périodes et en échéances. Υ est un ETPN et Π un ensemble d'algorithmes d'ordonnancement préemptifs, non-oisifs. Soit f une norme de $(\Upsilon,+,\cdot)$. f est une* ***norme valide****, ou un critère, de Σ si et seulement si :*

$$(\forall \tau \in \Upsilon), ((\exists Q \in \Pi, Q(\tau)) \Rightarrow f(\tau) \leq 1).$$

Ainsi, si un trafic est ordonnançable dans le référentiel Σ, alors toute norme valide rend une valeur inférieure ou égale à 1 sur ce trafic. Ceci conduit directement à la notion d'efficacité dans Σ.

Définition 38 - *[Hermant et al. 1996], [Leboucher 1998]. Soit $\Sigma = (\Upsilon, \Pi)$, un référentiel d'ordonnancement périodique compatible pour la faisabilité avec la propriété d'équivalence en périodes et en échéances. Υ est un ETPN et Π un ensemble d'algorithmes d'ordonnancement préemptifs, non-oisifs. Soit $P \in \Pi$ un algorithme d'ordonnancement donné, f une norme de $(\Upsilon,+,\cdot)$ et $V[\Upsilon]$ l'ensemble des normes valides de $(\Upsilon,+,\cdot)$. On appelle :*

- $\alpha_f(P) = sup\{\alpha / (\forall \tau \in \Upsilon, f(\tau) \leq \alpha \Rightarrow P(\tau))\}$ *la* ***f-efficacité*** *de P dans Σ*[1].
- $\varepsilon(P) = sup_{f \in V[\Upsilon]}\{\alpha_f(P)\}$ *l'****efficacité*** *de P dans Σ.*
- *f un* ***critère le plus fin*** *de Σ si et seulement si $\alpha_f = \varepsilon$.*

En d'autres termes $\varepsilon(P)$, l'efficacité de l'algorithme P dans le référentiel $\Sigma = (\Upsilon, \Pi)$ est la borne supérieure des *f*-efficacités, où *f* est le critère le plus fin dans l'ensemble $V[\Upsilon]$ des normes valides de $(\Upsilon,+,\cdot)$. Cette définition précise et remplace dorénavant la Définition 7 p. 11.

A priori le calcul de $\varepsilon(P)$ n'est pas trivial car cette définition conduit à tester toutes les normes valides sur tous les trafics. Dans certains cas cependant, l'analyse est considérablement simplifiée par l'identification d'un critère d'optimalité et d'un algorithme Σ-optimal (cf. Définition 24 p. 21). Des propriétés intéressantes peuvent alors être déduites. Nous en rappellons trois qui seront appliquées dans la suite. De nombreuses autres propriétés, telle que la non convexité des Σ-régions (cf. Définition 25 p. 21) en présence de priorités fixes, sont développées dans [Leboucher 1998].

Théorème 16 - *[Hermant et al. 1996], [Leboucher 1998]. Soit $\Sigma = (\Upsilon, \Pi)$ un référentiel d'ordonnancement périodique compatible pour la faisabilité avec la propriété d'équivalence en périodes et en échéances. Υ est un ETPN et Π un ensemble d'algorithmes d'ordonnancement préemptifs, non-oisifs. S'il existe $P_o \in \Pi$ qui vérifie $\alpha_{N_o}(P_o) = \varepsilon(P_o) = 1$, alors :*

- *(1) P_o est Σ-optimal,*
- *(2) N_o est un critère le plus fin qui est appelé le* ***critère d'optimalité****,*
- *(3) $\forall P \in \Pi$, $\alpha_{N_o}(P) = \varepsilon(P)$.*

Le principe des preuves est rappelé pour illustrer l'intérêt d'une telle approche.

Propriété (1). $(\alpha_{N_o}(P_0) = \varepsilon(P_0) = 1) \Rightarrow (P_0 \ est\ \Sigma - \text{optimal})$ par construction mais l'inverse sur cette implication n'est pas vrai. A titre d'exemple, si Υ contient tout TPN caractérisé par $D_i = T_i$, $\forall i \in [1, n]$ et Π contient des algorithmes préemptifs à priorités fixes mais pas *EDF*, alors *RM* est Σ-optimal mais nous ne connaissons pas de critère d'optimalité. Le meilleur minorant connu sur l'efficacité de *RM* vaut $\alpha_U(RM) = n(2^{1/n} - 1)$ (cf. [Liu and Layland 1973])[2].

1. $sup\{A\}$ désigne la borne supérieure de l'ensemble *A*.

2. Une conjecture est que *U* est un critère le plus fin dans ce référentiel. $n(2^{1/n} - 1)$ serait alors une mesure exacte de l'efficacité de *RM*, même lorsque *EDF* n'est pas dans Π.

Propriété (2). N_o étant un critère d'optimalité par hypothèse, vérifions que c'est bien un critère le plus fin. Soit $N \in V[\Upsilon]$ une norme valide quelconque. $\alpha_N \leq \alpha_{N_o}$ est équivalent par définition à :

$$(\forall P \in \Pi)(\forall \tau \in \Upsilon)(N_o(\tau) \leq \alpha_N(P) \Rightarrow P(\tau)).$$

Supposons que $N_o(\tau) \leq \alpha_N(P)$, alors $N_o(\tau / \alpha_N(P)) \leq 1$ puisque N_o est une norme. Comme par hypothèse $\alpha_{N_o}(P_o) = 1$, nous avons immédiatement $P_0(\tau / \alpha_N(P))$, c.à.d. que le TPN $\tau / \alpha_N(P)$ est faisable par P_o. Comme de plus N est une norme valide, alors d'après la Définition 37 p. 53, $N(\tau / \alpha_N(P)) \leq 1$. Nous avons donc $N(\tau) \leq \alpha_N(P)$ c.à.d. $P(\tau)$, le TPN τ est faisable par P. Finalement, comme α_{N_o} est plus grand que n'importe quel α_N, nous avons $\alpha_N \leq \alpha_{N_o} = \varepsilon = 1$ (N_o est un critère le plus fin).

A titre d'exemple, si Υ contient tout TPN caractérisé par $D_i = T_i$, $\forall i \in [1, n]$ et Π contient tout algorithme d'ordonnancement préemptif non-oisif, alors *EDF* est Σ-optimal, U est un critère d'optimalité et toute norme valide N vérifie $\alpha_N \leq \alpha_U$. En particulier (cf. Théorème 6 p. 36) :

$$(\forall \tau \in \Upsilon)(U(\tau) \leq n(2^{1/n} - 1) \Rightarrow RM(\tau)).$$

Propriété (3). Soit $P \in \Pi$ et N_{fc} un critère le plus fin. D'après la Définition 38 p. 53 :

$$\alpha_{N_{fc}}(P) = sup\{\alpha / (\forall \tau \in \Upsilon, N_{fc}(\tau) \leq \alpha \Rightarrow P(\tau))\} = \varepsilon(P) \geq \alpha_{N_o}(P).$$

Pour prouver $\alpha_{N_o}(P) = \varepsilon(P)$ il faut montrer que nous avons aussi $\alpha_{f_c}(P) \leq \alpha_{N_o}(P)$, c.à.d. :

$$(\forall \tau \in \Upsilon), (N_o(\tau) \leq \alpha_{N_{fc}}(P) \Rightarrow P(\tau)).$$

Comme N_{fc} est une norme valide, nous avons $(\forall \tau \in \Upsilon), (N_o(\tau) \leq 1 \Rightarrow N_{fc}(\tau) \leq 1)$ et donc :

$(\forall \tau \in \Upsilon), (N_o(\tau) \leq \alpha_{N_{fc}}(P) \Rightarrow N_{fc}(\tau) \leq \alpha_{N_{fc}}(P))$, c.à.d. $(\forall \tau \in \Upsilon), (N_o(\tau) \leq \alpha_{N_{fc}}(P) \Rightarrow P(\tau))$.

VI.1.2.2. Calculs exacts de l'efficacité

Le Théorème 12 p. 46, nous indique que *EDF* est optimal en présence de trafics non-concrets. D'après la propriété (3), il faut donc juste identifier un critère d'optimalité pour pouvoir mesurer l'efficacité des autres ordonnanceurs en présence de *EDF*.

Théorème 17 - *[Hermant et al. 1996], [Leboucher 1998]. Soit $\Sigma=(\Upsilon,\Pi)$, un référentiel d'ordonnancement périodique compatible pour la faisabilité avec la propriété d'équivalence en périodes et en échéances. Υ contient tout TPN et Π est un ensemble d'algorithmes. N_U et N_{EDF} sont des normes valides et N_{EDF} est un critère d'optimalité avec :*

- $N_U(\tau) = \sum_{(T,D) \in T \times D} \frac{|C_{T,D}|}{T}$,
- $N_{EDF}(\tau) = sup_{t \in \mathbb{R}^+} \left\{ \frac{1}{t} \sum_{(T,D) \in T \times D} max\left\{0, 1 + \left\lfloor \frac{t-D}{T} \right\rfloor\right\} |C_{T,D}| \right\}$.

N_U et N_{EDF} sont basées respectivement sur la charge et la demande processeur (cf. chapitre III.1. p. 15) et vérifient trivialement les trois propriétés d'une norme (cf. page 53). *EDF* étant Σ-optimal, tout trafic non-concret $\tau \in \Upsilon$ faisable dans Σ est donc *EDF*-faisable. Du Théorème 12 p. 46, L. Leboucher note alors que $N_{EDF}(\tau) \leq 1$ et de la Définition 38 p. 53, que $\alpha_{N_{EDF}}(EDF) = \varepsilon(EDF) = 1$. En faisant croître t à l'infini, nous avons aussi $N_U(\tau) \leq N_{EDF}(\tau)$.

N_{EDF} étant un critère d'optimalité, nous récupérons la propriété (3) c.à.d. que pour tout $P \in \Pi$, alors $\alpha_{N_{EDF}}(P) = \varepsilon(P)$. La demande processeur, jusqu'alors utilisée uniquement avec *EDF*, donne donc en présence de cet algorithme et de trafics non-concrets un calcul exact de l'efficacité dans un référentiel Σ de tout autre algorithme d'ordonnancement compatible pour la faisabilité avec la propriété d'équivalence en périodes et en échéances[1].

1. Si cette relation d'équivalence ne tient pas, il reste possible, sans recourir à l'algèbre linéaire, de calculer une borne de faisabilité basée sur la demande processeur (cf. chapitre VI.1. p. 51, pour une illustration dans le cas non-préemptif).

VI.1.3. Efficacité des algorithmes à priorités fixes

Le critère d'optimalité N_{EDF} généralise N_U, l'un des rares points de comparaison connus sur la performance de algorithmes (cf. chapitre VI.1.1. p. 51). Notons d'ailleurs que $N_{EDF}=N_U$ dans le cas $D_i = T_i$, $\forall i \in [1, n]$ (cf. Théorème 2 p. 30). Voyons comment il est possible d'étendre la comparaison en minorant l'efficacité des algorithmes à priorités fixes face à *EDF*.

Soit $\Sigma = (\Upsilon,\Pi)$, où Υ est caractérisé par des couples (T,D) et Π par des algorithmes compatibles pour la faisabilité avec la propriété d'équivalence en période et en échéance. *EDF* est Σ-optimal et N_{EDF} est un critère d'optimalité. Il suit que :

- l'efficacité de *EDF* est toujours maximale car $\alpha_{N_{EDF}}(EDF) = \varepsilon(EDF) = 1$,
- pour tout autre algorithme $P \in \Pi$, $\alpha_{N_{EDF}}(P) = \varepsilon(P) \leq \varepsilon(EDF)$,
- pour toute norme valide N, $\alpha_N(P) \leq \alpha_{N_{EDF}}(P) = \varepsilon(P) \leq 1$.

Si P est un algorithme à priorités fixes quelconque, nous allons maintenant utiliser la norme N_U pour établir le comportement asymptotique de $\varepsilon(P)$ face à différentes configurations de couples (T,D). En effet, N_U n'est pas un critère d'optimalité dans le cas général mais conduit à un calcul trivial d'un minorant de $\varepsilon(P)$. Afin d'améliorer la mesure, un théorème d'efficacité est proposé par L. Leboucher. Nous en rappelons le principe (cf. chapitre VI.1.3.1. p. 55) puis l'application proposée dans [Hermant et al. 1996] (cf. chapitre VI.1.3.2. p. 55). Nous spécialisons ensuite ce résultat pour *DM*, un algorithme à priorités fixes montrant un bon comportement face aux propriétés d'optimalité et de dominance (cf. chapitre VI.1.4. p. 58).

VI.1.3.1. Principe du théorème d'efficacité

L'objectif est d'établir en deux étapes une minoration permettant d'interpréter simplement $\varepsilon(P)$ dans un référentiel d'ordonnancement $\Sigma=(\Upsilon,\Pi)$ qui contient P_{opt}, un algorithme Σ-optimal et où l'utilisation du critère d'optimalité N_{opt} est coûteuse[1].

Soit N une norme valide simple quelconque, la première étape conduit à établir une condition **nécessaire** pour P_{opt} basée sur N. Cette condition peut être écrite : $\forall \tau \in \Upsilon$, $N(\tau) \leq \alpha$ avec $\alpha \geq \alpha_N(P_{opt})$. α est donc un majorant de la N-efficacité de P_{opt}. N/α est alors une norme valide puisque si τ est faisable par l'algorithme P_{opt}, alors la condition $N(\tau) \leq \alpha$ est nécessairement vérifiée (et donc $N(\tau)/\alpha \leq 1$). Il est intéressant de remarquer que cette norme est une meilleure approximation que N du critère d'optimalité N_{opt} lorsque $\alpha < 1$.

La deuxième étape conduit à établir une condition **suffisante** pour P, toujours basée sur N. Cette condition peut être écrite $\forall \tau \in \Upsilon$, $N(\tau) \leq \beta \leq \alpha_N(P)$ avec β un minorant de la N-efficacité de P. Il vient que $N(\tau)/\alpha \leq \beta/\alpha$ est une condition suffisante pour la P-faisabilité où N/α est la norme valide introduite en première étape. On en déduit que β/α est une minoration de l'efficacité de P et que celle ci est meilleure que β lorsque $\alpha < 1$.

VI.1.3.2. Théorème d'efficacité pour tout algorithme à priorités fixes

Soit $P \in \Pi$, un algorithme à priorités fixes compatible pour la faisabilité avec la propriété d'équivalence en période et en échéance. En appliquant le raisonnement précédent avec N_U voyons, avant de spécialiser ce résultat pour *DM*, comment L. Leboucher minore $\varepsilon(P)$ en comparaison de *EDF* et en fonction des couples (T,D) représentés dans Υ.

1. Le calcul de l'efficacité par N_{EDF} est relativement coûteux. Une procédure bornant ce coût est proposée dans [Hermant et al. 1996] et sera développée dans [Hermant 1998], [Leboucher 1998].

VI.1.3.2.a Etablissement du théorème

La première étape du raisonnement cherche à établir une norme valide $N'_U = N_U/\alpha$ à partir d'une condition nécessaire pour *EDF*. Le Théorème 12 p. 46, nous indique qu'un trafic $\tau \in \Upsilon$ est nécessairement faisable par *EDF* si :

$$\forall t \geq 0\,,\ t \geq \sum_{i=1}^{n} max\{0,\ 1 + \lfloor (t - max(D_i))/T_i \rfloor\} C_i\,.$$

En se limitant à $t = max(D_i)$, on a la condition nécessaire suivante, basée sur N_U, pour *EDF* :

$$(max(D_i) \geq \sum_{i \leq n} C_i) \Rightarrow \left(\frac{max(D_i)}{min(T_i)} \geq \frac{\sum_{i \leq n} C_i}{(min(T_i))} \geq \sum_{i \leq n} \frac{C_i}{T_i} = N_U \right). \qquad (19)$$

Ainsi lorsque $max(D_i) \leq min(T_i)$, un majorant sur la N_U-efficacité de *EDF* vaut donc $\alpha = max(D_i)/min(T_i)$. Pour tout $\tau \in \Upsilon$, si $N_{EDF}(\tau) \leq 1$, alors l'éq. (19) est nécessairement vérifiée, c.à.d. $N'_U(\tau) = N_U(\tau)/\alpha \leq 1$. Il vient que $N'_U = N_U/\alpha$ est une norme valide.

Par contre lorsque $max(D_i) \geq min(T_i)$, c.à.d. $\alpha \geq 1$, le majorant obtenu n'est pas intéressant puisque nous savons par définition que l'efficacité de tout algorithme est majoré par 1. Dans le cas général, nous avons la norme valide suivante.

$$N'_U = \frac{max(max(D_i), min(T_i))}{max(D_i)} N_U \qquad (20)$$

La deuxième étape du raisonnement cherche à établir β, un minorant de l'efficacité de *P* toujours basée sur N_U (c.à.d. une condition suffisante pour *P*). Tout d'abord, le Théorème 14 p. 50, permet de majorer comme suit les *level-i busy periods* de la tâche τ_i :

$$\forall i, \forall q\ ,\ w_{i,q} \leq (q+1)C_i + \sum_{j<i} (w_{i,q}/T_j + 1) C_j\ ,$$

$$\text{c.à.d.,}\quad \forall i, \forall q\ ,\ w_{i,q} - qT_i < \frac{qT_i(\sum_{j \leq i} C_j/T_j - 1) + \sum_{j \leq i} C_j}{1 - \sum_{j<i} C_j/T_j}\ .$$

Comme $r_i = max(w_{i,q} - qT_i) \leq D_i$ et $qT_i(\sum_{j \leq i} C_j/T_j - 1) \leq 0$ si le trafic est faisable par *P*, le test suivant est suffisant pour la faisabilité d'un trafic non-concret τ par *P* :

$$\forall i\ ,\ \sum_{j \leq i} C_j/(1 - \sum_{j<i} C_j/T_j) \leq D_i\,. \qquad (21)$$

Il est donc suffisant que : $max(T_i) \sum_{j \leq n} C_j/T_j/(1 - \sum_{j \leq n} C_j/T_j) \leq min(D_i)$,

$$\text{et donc suffisant que } N_U \leq min(D_i)/(max(T_i) + min(D_i))\,. \qquad (22)$$

$\beta = min(D_i)/(max(T_i) + min(D_i))$ est alors un minorant sur la N_U-efficacité de *P* qui conduit au théorème d'efficacité suivant.

Théorème 18 - *[Hermant et al. 1996], [Leboucher 1998]. Soit un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN et* Π *contient EDF et P, un algorithme d'ordonnancement préemptif à priorités fixes compatible pour la faisabilité avec la propriété d'équivalence en périodes et en échéances. Nous avons :*

$$\varepsilon(P) \geq \frac{max(max(D_i), min(T_i))}{max(T_i) + min(D_i)} \cdot \frac{min(D_i)}{max(D_i)}$$

Ce résultat est obtenu en associant l'éq. (22) et l'éq. (20). En reprenant le raisonnement du chapitre VI.1.3.1. p. 55, nous avons donc obtenu β/α, une minoration de l'efficacité de *P* meilleure que β lorsque $\alpha = max(D_i)/min(T_i) < 1$.

VI.1.3.2.b Interprétation du théorème d'efficacité

Le théorème précédent donne une minoration de l'efficacité des algorithmes préemptifs à priorités fixes. Cette-ci est pessimiste car Υ est approximé par $\delta T = max(T_i)/min(T_i)$, la dispersion en périodes, et $\delta D = max(D_i)/min(D_i)$, la dispersion en échéances relatives, des couples (T,D) dans Υ. Elle est de plus valable quelque soit l'algorithme *P* à priorités fixes choisi (elle couvre en particulier les plus mauvais). En dépit de ce pessimisme, *EDF* étant l'algorithme le plus efficace ($\varepsilon(EDF) = 1$ dans tous les cas, cf. Théorème 17 p. 54), ce résultat permet de lui comparer l'efficacité des algorithmes à priorités fixes en fonction des couples (T,D) représentés dans Υ. Nous en rappelons l'interprétation proposée dans [Hermant et al. 1996] et [Leboucher 1998] que nous spécialiserons ensuite pour l'algorithme *DM* (cf. chapitre VI.1.4. p. 58).

Le cas homogène (c.à.d. $\forall i$, $D_i=D$, $T_i=T$).

Dans ce cas trivial tout TPN est équivalent en période et en échéance à un TPN monotâche caractérisé par l'unique couple (T, D) dans Υ ayant une durée d'exécution $C = \sum C_i$). Le test de faisabilité est alors la même pour *EDF* et *P* : $C \leq T$ si $D \geq T$ (c.à.d. $N_U \leq 1$) ; $C \leq D$ si $D < T$ (c.à.d. $N_U \leq D/T$). Tout algorithme à priorités fixes est donc Σ-optimal et son efficacité vaut 1 (comme *EDF*). Notons que la N_U-efficacité de *DM* est égale à D/T dans le cas $D < T$. Ceci illustre bien que cette norme n'est pas un critère le plus fin. A titre de comparaison, le théorème d'efficacité minore $\varepsilon(P)$ par $T/(T + D) = 1/(1 + D/T)$ lorsque $D < T$, et par $D/(T + D)$ lorsque $D \geq T$. Il est ainsi possible d'évaluer l'erreur induite dans ce cas (50% si $D = T$).

Le cas de Liu & Layland (c.à.d. $\forall i$, $D_i = T_i$).

Dans ce cas, la dispersion $\delta D = \delta T$. Nous savons que $\alpha_{N_{U/n}}(P) = \varepsilon(P)$ (cf. chapitre VI.1.1. p. 51) et [Liu and Layland 1973] montrent que $\varepsilon(RM)$ vaut $n(2^{1/n} - 1)$. A titre de comparaison, le théorème d'efficacité minore $\varepsilon(P)$ par : $min(T_i)/(max(T_i) + min(T_i)) = 1/(\delta T + 1)$.

Si nous considérons que *n*, le nombre de tâches, peut être comparé à une forme de dispersion, il est remarquable que pour ces deux résultats la minoration sur l'efficacité de *P* augmente avec la dispersion[1]. Dans le cas particulier où *P=RM* (le meilleur algorithme à priorités fixes dans ce contexte), ce résultat est cependant bien moins fin que la borne de [Liu and Layland 1973].

Le cas $\{D_i\}$ » $\{T_i\}$ où toutes les échéances relatives sont supérieures aux périodes.

Dans ce cas, nous savons du Théorème 2 p. 30, que $\alpha_{N_U}(P) = \varepsilon(P)$. Le théorème d'efficacité minore $\varepsilon(P)$ par $min(D_i)/(max(T_i) + min(D_i))$. En d'autres termes, plus $Max(T_i)$ est petit face à $Min(D_i)$, plus $\varepsilon(P)$ est proche de 1. Il est en effet intuitivement normal que lorsque les échéances relatives sont grandes face aux périodes, le choix de l'algorithme perde de son importance.

Le cas $\{D_i\}$ « $\{T_i\}$ où toutes les échéances relatives sont inférieures aux périodes.

Dans ce cas, le théorème d'efficacité nous montre que $\varepsilon(P)$ est minorée par :

$$\frac{min(T_i)}{(max(T_i) + min(D_i))} \cdot \frac{(min(D_i))}{(max(D_i))}.$$

Ce qui donne $1/(\delta T \delta D)$ lorsque $min(D_i)$ devient négligeable devant $max(T_i)$. En d'autres termes, le minorant sur $\varepsilon(P)$ diminue d'autant plus que le $max(D_i)$ est proche du $min(T_i)$. Ceci est encore plus vrai lorsque la dispersion (en échéance ou en période) augmente.

Le cas général où les échéances relatives ne sont pas liées aux périodes.

Dans le cas général, il n'y a pas d'interprétation claire du théorème d'efficacité mais un minorant non nul sur $\varepsilon(P)$. La conjecture est que l'efficacité des algorithmes à priorités fixes baisse lorsque la dispersion augmente.

1. Notons cependant qu'il est possible d'avoir une dispersion faible avec un grand nombre de tâches.

VI.1.4. Efficacité de *DM*

Le chapitre V.2.1.2. p. 48, identifie *DM* comme un algorithme à priorités fixes intéressant. Celui-ci est Σ-optimal dans le cas $D_i \geq T_i$, $\forall i \in [1, n]$ face à tout autre algorithme à priorités fixes et nous ne connaissons pas d'algorithme qui assignerait les priorités fixes de façon optimale et à faible coût dans un référentiel plus large.

Nous allons maintenant spécialiser l'étude précédente de l'efficacité, valable pour tout algorithme à priorités fixes compatible pour la faisabilité avec la propriété d'équivalence en période et en échéance, pour *DM* uniquement.

L'objectif est de diminuer un peu les approximations afin d'améliorer l'interprétation de la minoration de $\varepsilon(DM)$, l'efficacité de *DM*, en fonction des couples (T,D) représentés dans Υ .

VI.1.4.1. Un théorème d'efficacité pour *DM*

Nous utilisons ici la charge partielle $U_i = \sum_{j \leq i} C_j/T_j$ au lieu de N_U.

En première étape nous établissons une condition nécessaire pour *EDF*, basée ici sur la charge partielle U_i. Pour tout trafic non-concret $\tau \in \Upsilon$, du Théorème 12 p. 46, nous avons la condition nécessaire suivante :

$$\forall i \qquad D_i \geq h_{D_j \leq D_i}(D_i) + h_{D_k > D_i}(D_i) = \sum_{D_j \leq D_i}(1 + \lfloor (D_i - D_j)/T_i \rfloor)C_i \geq \sum_{D_j \leq D_i} C_i ,$$

$$\text{c.à.d. la condition nécessaire : } \forall i \qquad \frac{D_i}{min_{D_j \leq D_i}(T_j)} \geq \frac{\sum_{D_j \leq D_i} C_i}{min_{D_j \leq D_i}(T_j)} .$$

Si les indices sont attribués aux tâches par ordre d'échéances relatives croissantes, c.à.d. dans l'ordre *DM* qui nous intéresse ici (cf. chapitre IV.3.3.2. p. 37), alors cette condition nécessaire est équivalente à :

$$\forall i \qquad \frac{D_i}{min_{j \leq i}(T_j)} \geq \frac{\sum_{j \leq i} C_i}{min_{j \leq i}(T_j)} \geq U_i .$$

Si τ est faisable par *EDF*, l'équation précédente est nécessairement vérifiée. En la combinant avec la condition nécessaire triviale, $\forall i$, $U_i \leq 1$, nous obtenons finalement la condition nécessaire :

$$\forall i \qquad U'_i = \frac{max(D_i, min_{j \leq i}(T_j))}{D_i} U_i \leq 1 . \tag{23}$$

En deuxième étape nous cherchons à établir une condition suffisante pour *DM,* toujours basée sur U_i. De l'équation (21) p. 56, nous avons la condition suffisante suivante :

$$\forall i \qquad \frac{max_{j \leq i}(T_j) \sum_{j \leq i} C_j/T_j}{1 - \sum_{i \leq i} C_j/T_j + C_i/T_i} \leq D_i ,$$

$$\text{c.à.d. la condition suffisante : } \forall i \qquad U_i \leq \frac{D_i(1 + C_i/T_i)}{max_{j \leq i}(T_j) + D_i} . \tag{24}$$

Nous en déduisons le théorème d'efficacité suivant pour *DM*.

Théorème 19 - *Soit un référentiel d'ordonnancement* $\Sigma = (\Upsilon, \Pi)$ *où* Υ *contient tout TPN et* Π *contient EDF et DM. Nous avons :*

$$\varepsilon(DM) \geq min_i \left\{ \frac{max(D_i, min_{j \leq i}(T_j))}{max_{j \leq i}(T_j) + D_i} \right\} \quad (25)$$

Preuve. En combinant l'équation (23) et l'équation (24), nous avons la condition suffisante suivante pour *DM* :

$$\forall i \qquad U_i = \frac{D_i \cdot U'_i}{Max(D_i, Min_{j \leq i}(T_j))} \leq \frac{D_i(1 + C_i/T_i)}{Max_{j \leq i}(T_j) + D_i},$$

$$\text{c.à.d. la condition suffisante, } max_i(U'_i) \leq min_i \left\{ \frac{max(D_i, min_{j \leq i}(T_j))}{max_{j \leq i}(T_j) + D_i} \right\}. \quad (26)$$

Si τ est faisable par *EDF*, $max_i(U'_i) \leq 1$ est vérifié par l'équation (23) qui est une condition nécessaire pour *EDF*. $max_i(U'_i)$ est donc une norme valide.

L'équation (26) est une condition suffisante pour *DM* basé sur la norme valide $max_i(U'_i)$. Elle nous donne donc une minoration de $\varepsilon(DM)$ dans Σ. ❐

VI.1.4.2. Interprétation du théorème d'efficacité pour *DM*

La minoration de $\varepsilon(DM)$ précédente nous permet de spécialiser l'interprétation asymptotique du chapitre VI.1.3.2. p. 55, pour cet algorithme d'ordonnancement. En particulier l'approximation de Υ par la notion de dispersion en échéance relative $\delta D = max(D_i)/min(D_i)$ disparaît etla notion de dispersion en périodes $\delta T = max(T_i)/min(T_i)$ est affinée puisqu'elle considère un sous-ensemble des couples *(T,D)*.

Si les indices sont attribués aux couples (T,D) par ordre d'échéances relatives croissantes, c.à.d. dans l'ordre *DM*, voyons maintenant comment il est possible d'interpréter cette minoration de $\varepsilon(DM)$ en fonction de couples (T,D) représentés dans Υ.

Le cas homogène (c.à.d. $\forall i$, $D_i=D$, $T_i=T$).

L'interprétation est identique dans ce cas à celle proposée lors du chapitre VI.1.3.2. p. 55.

Le cas de Liu & Layland (c.à.d. $\forall i$, $D_i = T_i$).

Dans ce cas nous avons la minoration suivante, qui annule l'influence forte de la dispersion sur l'efficacité constatée lors du chapitre VI.1.3.2. p. 55, :

$$\varepsilon(DM) \geq min_i \left\{ \frac{max(T_i, min_{j \leq i}(T_j))}{max_{j \leq i}(T_j) + T_i} \right\} = min_i \left\{ \frac{T_i}{(T_i + T_i)} \right\} = 1/2 .$$

Nous savons que *DM=RM* est l'algorithme d'ordonnancement à priorités fixes optimal dans ce cas. Ce minorant approxime donc mieux $n(2^{1/n} - 1)$, la borne de faisabilité de [Liu and Layland 1973] avec *RM*, pour laquelle seul *n* (le nombre de tâches) influe en tendant rapidement vers $log_e 2$ quand *n* tend vers l'infini.

Tout en étant plus pessimiste que cette borne, le minorant obtenu est cependant plus général comme le montre l'étude des autres cas.

Le cas $\forall i, D_i \geq min_{j \leq i}(T_j)$.

Dans ce cas, nous avons la minoration suivante qui ne peut diminuer si, à périodes constantes, les échéances augmentent :

$$\varepsilon(DM) \geq min_i \left\{ \frac{1}{\left(1 + \frac{max_{j \leq i}(T_j)}{D_i}\right)} \right\}.$$

Le cas $\forall i, D_i \geq T_i$.

Ce cas est inclus dans le précédent puisque sous l'hypothèse posée et avec l'algorithme *DM,* pour toute tâche d'indice *i* nous avons $D_i \geq D_{j \leq i} \geq T_j$. Pour la même raison le minorant obtenu sur $\varepsilon(DM)$ est alors supérieur à 1/2.

L'intérêt d'une telle constante est d'être indépendante de la dimension de la base de l'espace vectoriel $(\Upsilon, +, \cdot)$, dès lors que $\forall i, D_i \geq T_i$. Celle-ci peut être comparée à la borne $n(2^{1/n} - 1)$ obtenue par [Liu and Layland 1973] dans le cas $\forall i, D_i = T_i$ avec *RM.*

Le cas $\{D_i\}$ » $\{T_i\}$.

Ce cas est inclus dans le précédent, c.à.d. $\forall i, D_i \geq T_i$ et donc dans $\forall i, D_i \geq min_{j \leq i}(T_j)$. Le minorant sur $\varepsilon(DM)$ tend alors vers 1 puisque $max(T_i)$ devient petit devant $min(D_i)$.

La conclusion du chapitre VI.1.3.2. p. 55, dans le cas $\{D_i\}$ » $\{T_i\}$, est donc affinée pour *DM* puisque le comportement observé (c.à.d. la diminution de l'écart avec l'efficacité de *EDF* lorsque les échéances augmentent) se produit ici dès que pour toute tâche *i,* nous avons $D_i \geq min_{j \leq i}(T_j)$ et ne peut passer en dessous de 1/2 dès que pour toute tâche *i,* nous avons $\forall i, D_i \geq T_i$.

Les cas $\forall i, D_i \leq min_{j \leq i}(T_j)$ et $\{D_i\}$ « $\{T_i\}$.

Avec *DM*, ces deux cas sont équivalents. Le minorant sur $\varepsilon(DM)$ vaut :

$min_i \left\{ \frac{min_{j \leq i}(T_j)}{(max_{j \leq i}(T_j) + D_i)} \right\}$, obtenu pour $i=n$: $\frac{min(T)}{(max(T) + D_n)}$ (avec $D_n = max(D)$).

Dans ce cas le minorant sur $\varepsilon(DM)$ conduit à $1/\delta T$ (avec δT, la dispersion en périodes) lorsque D_n (la plus grande échéance relative) devient négligeable devant $max(T)$. Il tend donc vers 1 lorsque de plus δT diminue. L'intuition est donc confirmée puisque ces deux facteurs cumulés conduisent à une période occupée synchrone n'ayant qu'une activation par tâche. L'ordonnancement par *DM* et par *EDF* de cette période occupée (la pire en terme de faisabilité) est alors identique quelle que soit la dispersion en échéance.

Notons que cette augmentation de l'efficacité, bien que triviale, ne pouvait être détectée par la minoration précédente qui tenait compte de tous les algorithmes à priorités fixes. Les conclusions du chapitre VI.1.3.2. p. 55, dans les cas homogènes et $\{D_i\}$ » $\{T_i\}$, sont donc affinées pour *DM*. D'une part, la dispersion en échéance ne joue plus de rôle. D'autre part, le minorant sur $\varepsilon(DM)$ tend vers $1/\delta T$ dans ce cas.

Le cas général où les échéances relatives ne sont pas liées aux périodes.

Dans le cas général il n'y a pas d'interprétation claire du théorème d'efficacité mais un minorant non nul sur $\varepsilon(DM)$. La conjecture est donc la même que lors du chapitre VI.1.3.2. p. 55, mais nous avons pu préciser quelques cas où l'efficacité de *DM* ne peut passer sous la valeur 1/2 et/ou tend vers 1.

VI.2. Coût des conditions de faisabilité

En établissant notre cahier des charges lors du chapitre II.2. p. 8, nous avons noté que les conditions de faisabilité étaient dépendantes de l'algorithme d'ordonnancement utilisé. Au delà de la mesure de l'efficacité des algorithmes, nous avons donc décidé de nous intéresser à l'ordre de grandeur du coût des tests nécéssaires et suffisants associés (cf. Propriété 4 p. 12). L'état de l'art montre que ces tests sont pseudo-polynomiaux mais ne donne pas véritablement de points de comparaison. Nous rappelons donc brièvement les majorants obtenus dans [Hermant et al. 1996][1].

VI.2.1. Etat de l'art

VI.2.1.1. Faisabilité seule avec *EDF*

D'après le Théorème 4 p. 32, il faut tester le prédicat $t \le h(t)$ sur un intervalle borné pour établir la faisabilité d'un trafic par *EDF*. L'évaluation de ce prédicat est en *O(n)* et, dans le cas $D_i \le T_i$, $\forall i \in [1, n]$, [Baruah et al. 1990] et [Baruah et al. 1990b] établissent la borne suivante sur l'intervalle d'étude : $\sum_i (1 - D_i/T_i)C_i/(1 - U)$ (cf. Théorème 3 p. 32).

Par une manipulation algébrique supplémentaire, et en posant $U \le c < 1$, il est facile de montrer que cette approche conduit à un test pseudo-polynomial puisqu'elle permet de tester le prédicat $h(t) \le t$, en majorant la valeur maximale de *t* par :

$$\sum_i (1 - D_i/T_i)C_i/(1 - U) \le (c/(1 - c)) \cdot max(T_i - D_i).$$

En présence de trafics généraux, [Shin and Zheng 1994] étendent cette approche en tenant compte de $max_i(D_i)$ (cf. Théorème 4 p. 32). La manipulation algébrique précédente nous conduit alors à borner la valeur de *t* par :

$$max\left\{max_i(D_i),\ \sum_i (1 - D_i/T_i)C_i/(1 - U)\right\} \le max\{max_i(D_i),(c/(1 - c)) \cdot max(T_i - D_i)\}$$

VI.2.1.2. Pires temps de réponse avec un algorithme à priorités fixes

L'analyse de complexité faite dans la littérature est brève. D'après le Théorème 9 p. 39, $\forall i \in [1, n]$ il faut calculer *q* (avec $q \le Q_i$) *level-i busy periods* $w_{i,q}$ pour établir le pire temps de réponse de la tâche τ_i d'un trafic général par un algorithme d'ordonnancement à priorité fixes (avec $w_{i,Q_i} \le (Q_i + 1)T_i$). Comme chaque itération de l'équation récursive $w_{i,q}$ est en *O(n)*, que $w_{i,q} \le L_i \le L$ (cf. Lemme 2 p. 17) et que $L \le \sum_i C_i/(1 - c)$ si $U \le c < 1$ (cf. Lemme 4 p. 18), ce test nécéssaire et suffisant est donc pseudo-polynomial.

VI.2.1.3. Pires temps de réponse avec *EDF*

De même d'après le chapitre IV.2.4. p. 33, $\forall i \in [1, n]$ il faut calculer *a* (avec $a \le L$) *deadline(a+D_i) busy periods* $L_i(a)$ pour établir le pire temps de réponse de la tâche τ_i d'un trafic général par *EDF*. Comme chaque itération de l'équation récursive $L_i(a)$ est en *O(n)*, que $L_i(a) \le L$ (cf. Lemme 2 p. 17) et que $L \le \sum_i C_i/(1 - c)$ si $U \le c < 1$ (cf. Lemme 4 p. 18), [Spuri 1996] en déduit que cet test nécéssaire et suffisant est pseudo-polynomial.

1. développés par notre co-auteur J. F. Hermant (cf. [Hermant 1998]).

VI.2.2. Majorations

VI.2.2.1. Introduction

Nous simplifions ici les résultats de [Hermant et al. 1996] en nous focalisant sur une majoration du coût des tests nécéssaires et suffisants établis lors du chapitre V. p. 41.

A part une optimisation (dite d'accélération de convergence), la procédure de calcul considérée pour établir ces résultats est basée sur l'implémentation directe des tests énoncés au chapitre V. p. 41. Nous posons $U \leq c < 1$, $\delta T = max(T_i)/min(T_i)$ et ξ le coût d'une opération élémentaire (addition, multiplication, fonction) et utilisons systématiquement L, la taille de la période occupée synchrone du processeur, comme borne sur les intervalles d'étude. Avec les algorithmes à priorités fixes, il serait possible d'utiliser la borne L_i par tâche τ_i. Cependant :

- l'ordre de grandeur qui nous intéresse sur le coût des tests n'est pas modifié et nous avons montré lors du chapitre V.1.1. p. 41, que des bornes similaires (mais dont une analyse plus complète reste à faire) existent avec *EDF*.
- en toute rigueur, l'assignation de priorité hors-ligne optimale avec les algorithmes à priorités fixes en présence de trafics généraux est obtenue par la procédure d'*Audsley* dont la prise en compte mutiplierait par $O(n^2)$ la complexité des résultats. En pratique cependant, et à moindre coût, il est possible d'utiliser une assignation de priorités fixes intéressante telle que *DM* (cf. chapitre V.2.1.2. p. 48).

VI.2.2.2. Majorant sur le coût du calcul de *L*

Nous avons vu que l'équation (1) p. 17 ($L^{(m+1)} = W(L^{(m)})$) converge en un nombre fini d'itérations si $U \leq c < 1$ (cf. Lemme 3 p. 18). Appelons C_L le coût de calcul de L. D'après le Lemme 4 p. 18, C_L est majoré par $4n^2\lceil (c/(1-c)) \cdot \delta T \rceil \xi$.

Si la récursion s'arrête dès la première itération, c.à.d. lorsque $L^{(1)} = \sum_i C_i \leq min_i\{T_i\}$, C_L vaut $n\xi$ (plus précisément *n-1* additions). Plus généralement, si $\delta T = max(T_i)/min(T_i)$ et $U \leq c < 1$, C_L est pseudo-polynomial avec $C_L \leq O(n^2)$.

VI.2.2.3. Majorant sur le coût de calcul de la faisabilité seule par *EDF*

D'après le Théorème 12 p. 46, $\forall t \in S$ il faut tester le prédicat $h(t) \leq t$ sur l'intervalle $[0, L[$ pour établir la faisabilité d'un trafic par *EDF*. Pour être rigoureux $C_{EDF, (\forall t \in S, P(t))}$, le coût d'un tel calcul, intègre aussi C_L, le coût du calcul hors-ligne de la borne L.

$|S|$, le cardinal de S, correspond au nombre de points de discontinuités pertinents (les échéances absolues générées par le scénario d'activation synchrone) pour $h(t)$ dans l'intervalle $[0, L[$. Nous avons donc $|S| = \sum_i max(0, \lceil (L - D_i)/T_i \rceil)$.

Un majorant possible sur $|S|$, vaut $\sum_i \lceil L/T_i \rceil$, c.à.d. le nombre d'activations de tâches dans l'intervalle $[0, L[$. En établissant le Lemme 4 p. 18, nous avons vu que :

$$L \leq (c \cdot max(T_i))/(1-c) \text{ et donc } \sum_i \lceil L/T_i \rceil \leq n\lceil (c/(1-c)) \cdot \delta T \rceil,$$

Le coût de calcul du prédicat $h(t) \leq t$ vaut $7n\xi$ (*3n-1* additions, *2n+1* multiplications, *2n* fonctions). Nous avons donc le majorant suivant sur le coût de tous les prédicats à tester :

$$|S| \cdot 7n\xi \leq 7n^2\lceil (c/(1-c)) \cdot \delta T \rceil \xi$$

Notons que $|S|$ vaut 0 lorsque $L \leq min\{D_i\}$. Plus généralement, si $\delta T = max(T_i)/min(T_i)$, $U \leq c < 1$ et en intégrant C_L dans les calculs précédents, $C_{EDF, (\forall t \in S, P(t))}$ est donc pseudo-polynomial avec : $C_{EDF, (\forall t \in S, P(t))} \leq O(n^2)$.

VI.2.2.4. Majorant sur le coût du calcul des r_i par un algorithme à priorités fixes

D'après le Théorème 14 p. 50, $\forall i \in [1, n]$ il faut tester le prédicat $r_i = max_{q \leq Q_i}\{r_{i,q}\} \leq D_i$ (avec $r_{i,q}=w_{i,q}-qT_i$) pour calculer le pire temps de réponse de la tâche τ_i d'un trafic général par un algorithme d'ordonnancement à priorité fixes.

Appelons $C_{FP, (\forall i \in [1, n], r_i \leq D_i)}$, le coût global du test et notons que le calcul $Q_i = \lceil w_{i, Q_i}/T_i \rceil$ s'effectue en-ligne. Il n'est donc pas utile de prendre en compte le coût du calcul hors-ligne des w_{i, Q_i} dans $C_{FP, (\forall i \in [1, n], r_i \leq D_i)}$.

Le coût d'une itération ($w_{i, q}^{(k+1)} = (q+1)C_i + \sum_{j \in hp(i)} \lceil w_{i, q}^{(k)}/T_j \rceil C_j$) est de l'ordre de $4i\xi$ (i additions, $2i-1$ multiplications et $i-1$ fonctions si les tâches sont triées par priorités décroissantes).

Comme par définition $w_{i, q} \leq w_{i, q+1}$, nous pouvons accélérer la convergence du calcul récursif de $w_{i, q+1}$ en démarrant la suite sur $w_{i, q}$. En conséquence, si $|I|$ est, pour la tâche τ_i, le nombre d'itérations induit par les calculs de toutes les *level-i busy period* $w_{i, q}$ pertinentes pour calculer r_i (c.à.d. pour tout $q \leq Q_i$), alors $|I|$ est égal au nombre d'itérations pour calculer w_{i, Q_i} en initialisant le calcul à $w_{i, Q_i}^{(1)} = \sum_{j \in hp(i) \cup \{i\}} C_j$.

Pour la tâche d'indice i, un majorant possible sur $|I|$ vaut $\sum_{j \in hp(i) \cup \{i\}} \lceil w_{i, Q_i}/T_j \rceil$, c.à.d. le nombre d'activations de tâches de priorités supérieures ou égales à i dans l'intervalle $[0, w_{i, Q_i}[$. Du Lemme 2 p. 17 et du Lemme 4 p. 18, nous déduisons :

$$\sum_{j \in hp(i) \cup \{i\}} \lceil w_{i, Q}/T_j \rceil \leq \sum_i \lceil L/T_i \rceil \leq n \lceil (c/(1-c)) \cdot \delta T \rceil .$$

Pour toutes les tâches, nous obtenons donc le majorant suivant sur $C_{FP, (\forall i \in [1, n], r_i \leq D_i)}$ (en toute rigueur, il faudrait intégrer le surcoût nécessaire pour en déduire les r_i, mais cela ne modifie pas l'ordre de grandeur qui nous intéresse) :

$$C_{FP, (\forall i \in [1, n], r_i \leq D_i)} = \sum_i |I| \cdot 4i\xi \leq 4n^3 \lceil (c/(1-c)) \cdot \delta T \rceil \xi .$$

Notons que si pour toute tâche $|I|=1$ c.à.d. si $\forall i, Q_i = 0$ (la récursion s'arrête dès la première itération), nous avons alors $\sum_i |I| \cdot 4i\xi \approx 2n^2\xi$. Plus généralement, si $U \leq c < 1$ et $\delta T = max(T_i)/min(T_i)$, $C_{EDF, (\forall t \in S, P(t))}$ est donc pseudo-polynomial avec :

$$C_{FP, (\forall i \in [1, n], r_i \leq D_i)} \leq O(n^3) .$$

VI.2.2.5. Majorant sur le coût du calcul des r_i par *EDF*

D'après le Théorème 13 p. 47, $\forall i \in [1, n]$ il faut tester le prédicat $r_i = max_a\{r_i(a)\} \leq D_i$ (avec $r_i(a)=L_i(a)-a$) pour calculer les pires temps de réponse des tâches d'un trafic général par *EDF*. Appelons $C_{EDF, (\forall i \in [1, n], r_i \leq D_i)}$ le coût global du test. Seules les valeurs de a (en calant $a+D_i$ sur un point de discontinuité) dans l'intervalle *[0,L]* sont pertinentes. En toute rigueur $C_{EDF, (\forall i \in [1, n], r_i \leq D_i)}$ intègre donc aussi C_L, le coût du calcul hors-ligne de la borne L.

Le coût d'une itération de $L_i(a)$ (cf. équation (17) p. 46) est de l'ordre de $11n\xi$ (*4n-3* additions, *3n-1* multiplications, *4n-3* fonctions).

Comme $L_i(a_1) \leq L_i(a_2)$ si $a_1 \leq a_2$ (cf. [Hermant et al. 1996]), nous pouvons accélérer la convergence du calcul de $L_i(a_2)$ en démarrant la suite sur $L_i(a_1)$. En conséquence si $|I|$ est, pour la tâche τ_i, le nombre d'itérations induit par le calcul de toutes les *deadline busy period* $L_i(a)$ pertinentes pour calculer r_i, alors $|I|$ est égal au nombre d'itérations pour calculer $L_i(max\{a + D_i \leq L\})$ en initialisant la suite à $L_i^{(1)}(max\{a + D_i \leq L\}) = 0$.

Pour chaque tâche, un majorant sur $|I|$, vaut $\sum_i \lceil L/T_i \rceil$, c.à.d. le nombre d'activations de tâches dans l'intervalle $[0, L[$. En établissant le Lemme 4 p. 18, nous avons vu que:

$$L \leq (c \cdot max(T_i))/(1-c) \text{ et donc } \sum_i \lceil L/T_i \rceil \leq n\lceil (c/(1-c)) \cdot \delta T \rceil .$$

Pour toutes les tâches, nous obtenons le majorant suivant sur le coût de calcul de toutes les *deadline busy periods* $L_i(a)$ (en toute rigueur, il faudrait intégrer le surcoût nécessaire pour en déduire les r_i, mais cela ne modifie pas l'ordre de grandeur):

$$\sum_i |I| \cdot 11n\xi \leq 11n^3\lceil (c/(1-c)) \cdot \delta T \rceil \xi .$$

Notons que si $L \leq min\{T_i\}$, la récursion sur le calcul de toutes les *deadline busy periods* $L_i(a)$ s'arrête dès la deuxième itération, $|I| = 2$ (la récursion ne peut s'arrêter sur la première itération car nous initialisons le calcul des $L_i^{(1)}(a)$ à 0). Le calcul est alors en n^2. Plus généralement si $U \leq c < 1$ et $\delta T = max(T_i)/min(T_i)$, $C_{EDF,\,(\forall i \in [1,\,n],\, r_i \leq D_i)}$ est donc pseudo-polynomial (il faut y intégrer le C_L, le coût de calcul de L, mais cela ne modifie pas l'ordre de grandeur) avec :

$$C_{EDF,\,(\forall i \in [1,\,n],\, r_i \leq D_i)} \leq O(n^3) .$$

Partie C : Ordonnancement temps réel centralisé non-préemptif

Cette partie est consacrée à l'ordonnancement centralisé non-préemptif, non-oisif (c.à.d. sans inactivité possible du serveur lorsque sa file d'attente n'est pas vide, cf. Hypothèse 2 p. 10) en présence de trafics non-concrets τ. Ce problème, moins étudié que le cas préemptif, présente avec celui-ci des similitudes mais aussi quelques différences que nous allons tenter d'identifier. En particulier, nous allons voir que les propriétés d'optimalité et d'efficacité sont largement perturbées par les inversions de priorités causées par l'absence de préemption. La démarche suivie sera la même que dans le cas préemptif en y faisant référence dès que possible.

Plus précisément, le chapitre VII. p. 68, commence par examiner les résultats connus en termes de faisabilité, optimalité et pires temps de réponse. Ces résultats étant plus limités que dans le cas préemptif, le chapitre VIII. p. 72 propose ensuite quelques extensions issues de [George et al. 1996]. Le chapitre IX. p. 85, donne enfin des éléments de comparaison avec les résultats préemptifs.

Une synthèse sera proposée lors de la Partie D au regard des propriétés posées dans notre cahier des charges (cf. chapitre II.2.2. p. 10). L'impact de l'absence de préemption y sera illustré par quelques applications numériques.

VII. Etat de l'art

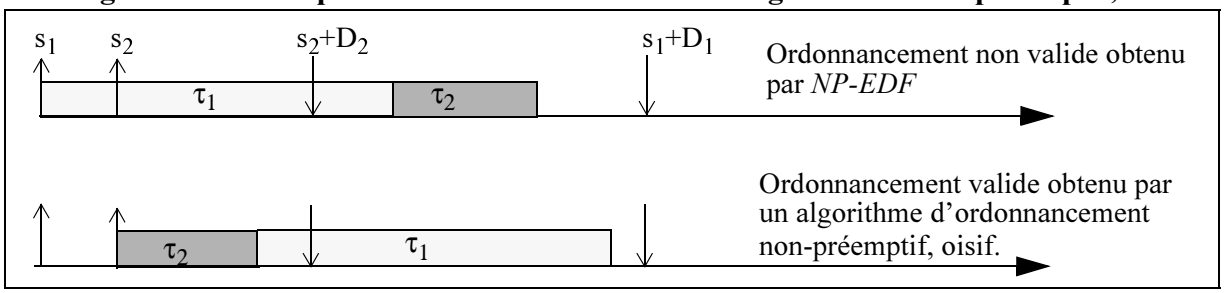
VII.1. Introduction

Comme pour le cas préemptif, le principal objectif des travaux existants est de coupler un algorithme d'ordonnancement optimal avec un test nécéssaire et suffisant pseudo-polynomial, basé ou non sur le calcul des pires temps de réponse des tâches. Avant d'examiner ces résultats, discutons rapidement de l'adéquation du comportement oisif/non-oisif des algorithmes non-préemptifs à notre cahier des charges.

VII.1.1. Algorithmes d'ordonnancement oisifs/non-oisifs

Ce problème ne se pose pas si le référentiel d'ordonnancement Σ contient uniquement des algorithmes préemptifs (oisifs ou non-oisifs). Les résultats d'optimalité, généralement établis avec des algorithmes préemptifs non-oisifs, restent valables en présence d'algorithmes préemptifs oisifs. Ceci n'est plus vrai si Σ contient uniquement des algorithmes non-préemptifs (oisifs ou non-oisifs). A titre d'exemple, la version non-préemptive et non-oisive de *EDF* (notée *NP-EDF* dans la suite) peut conduire à un ordonnancement non valide d'un trafic concret alors qu'il existe pourtant une solution non-préemptive, oisive valide (cf. Figure 11).

Figure 11 : Sous optimalité de NP-EDF face à un algorithmes non-préemptif, oisif



[Garey and Johnson 1979] montrent que l'établissement de la faisabilité d'un trafic concret est un problème NP-complet en contexte non-préemptif, oisif. Des propriétés optimales ont été identifiées (telle la décomposition du trafic initial en sous-trafics n'interférant pas les uns avec les autres [Yuan 1991], [Agrawala et al. 1994]) sans réduire pour autant la complexité pire cas. Au delà des problèmes de coût, ces résultats s'appuient sur des hypothèses de clairvoyance incompatibles (cf. Hypothèse 2 p. 10) avec notre cahier des charges, puisqu'ils imposent la connaissance des instants d'activation futurs pour prendre les décisions d'ordonnancement valides. Dans le cas des trafics non-concrets τ qui nous intéressent, [Howell et al. 1995] prouvent qu'il ne peut exister d'algorithme optimal non-préemptif oisif non-clairvoyant.

Ces inconvénients disparaissent si le référentiel d'ordonnancement est restreint aux algorithmes non-préemptifs non-oisifs et aux trafics non-concrets τ. Bien que dominés théoriquement par des solutions clairvoyantes oisives (ou éventuellement non-clairvoyantes oisives, ce qui est une question ouverte), des résultats d'optimalité pratiques peuvent toutefois être établis et accompagnés de calculs pseudo-polynomiaux de faisabilité et de pires temps de réponse. Rappelons en effet que la majorité des systèmes opérationnels sont non-oisifs

VII.1.2. Résultats existant

Les notions et les résultats proposés dans le cas non-préemptif non-oisif s'inspirent du cas préemptif. Ils ont fait cependant l'objet à notre connaissance de moins de publications.

Table 4 : Résultats non-préemptifs non-oisifs

	priorités dynamiques (cf. chapitre VII.2. p. 69)	priorités fixes (cf. chapitre VII.3. p. 71)
Optimalité	[Kim and Naghibdadeh 1980], [Jeffay et al. 1991]	
Tests donnant la faisabilité seule	[Kim and Naghibdadeh 1980], [Jeffay et al. 1991], [Shin and Zheng 1994]	
Tests donnant la faisabilité et les pires temps de réponse		[Hansson et al. 1994], [Burns et al. 1995]

Les algorithmes d'ordonnancement considérés sont *HPF* non-préemptifs, non-oisifs et se distinguent par leurs techniques d'assignation de priorités (cf. Définition 8 p. 13). Nous rappelons[1] brièvement les résultats en notre connaissance pour les priorité dynamiques puis pour les priorités fixes. Bien que cela ne fasse pas partie de l'état de l'art, nous utiliserons à nouveau le concept de référentiel d'ordonnancement[2] (cf. Définition 21 p. 20) afin de lever toute ambiguïté sur l'énoncé des résultats. Quelques généralisation/discussions seront ensuite proposées lors du chapitre VIII. p. 72, pour les case grisées (en particulier pour les cases vides).

VII.2. Priorités dynamiques

La littérature considère plus particulièrement la version Non-Préemptive, Non-Oisive de *EDF* (notée ***NP-EDF*** dans la suite). La différence avec le cas préemptif est qu'à tout instant une inversion de priorité peut se produire, résultant de l'arrivée d'une activation de tâche ayant une échéance absolue inférieure à celle de l'activation de tâche en cours d'exécution.

Contrairement au cas préemptif, l'optimalité de *NP-EDF* est prouvée **uniquement** par l'examen de quelques cas particuliers de faisabilité de trafics non-concrets τ. Le calcul des pires temps de réponse des tâches n'est pas considéré dans la littérature.

Théorème 20 - *[Kim and Naghibdadeh 1980], [Jeffay et al. 1991]. Soit un référentiel d'ordonnancement* $\Sigma=(\Upsilon,\Pi)$ *ou* Υ *contient tout TPN caractérisé par* $D_i = T_i \ \forall i \in [1,n]$ *et* Π *contient tout algorithme d'ordonnancement non-préemptif non-oisif. NP-EDF est* Σ*-optimal et* $\forall(\tau \in \Upsilon)$, τ *est NP-EDF-faisable si et seulement si:*

$$U \leq 1 \ et \ \forall t \in [0, max\{D_i\}] , \quad t \geq h(t) + max_{D_i > t}\{C_i - 1\} \qquad (27)$$

$$avec \ max_{D_i > t}\{C_i - 1\} = 0 \ si \ \nexists i : D_i > t.$$

Cette condition étend le résultat de [Liu and Layland 1973]. Elle est trivialement nécessaire pour tout algorithme non-préemptif, non-oisif en présence de trafics non-concrets (cf. Lemme 16 p. 72 pour une généralisation) et est aussi suffisante pour *NP-EDF* (cet algorithme est donc Σ-optimal dans ce cas). Supposons en effet qu'elle soit vérifiée et qu'il existe pourtant un scénario d'activation ω ordonnancé par *NP-EDF* qui rate une échéance, alors il est possible de durcir ce scénario (cf. Figure 12) pour obtenir l'un des scénarii testé par l'éq.

1. Nous sommes en temps discret (cf. Hypothèse 3 p. 10). Les inversions de priorités ne seront donc visibles qu'une unité de temps après le début d'exécution des activations de tâches.

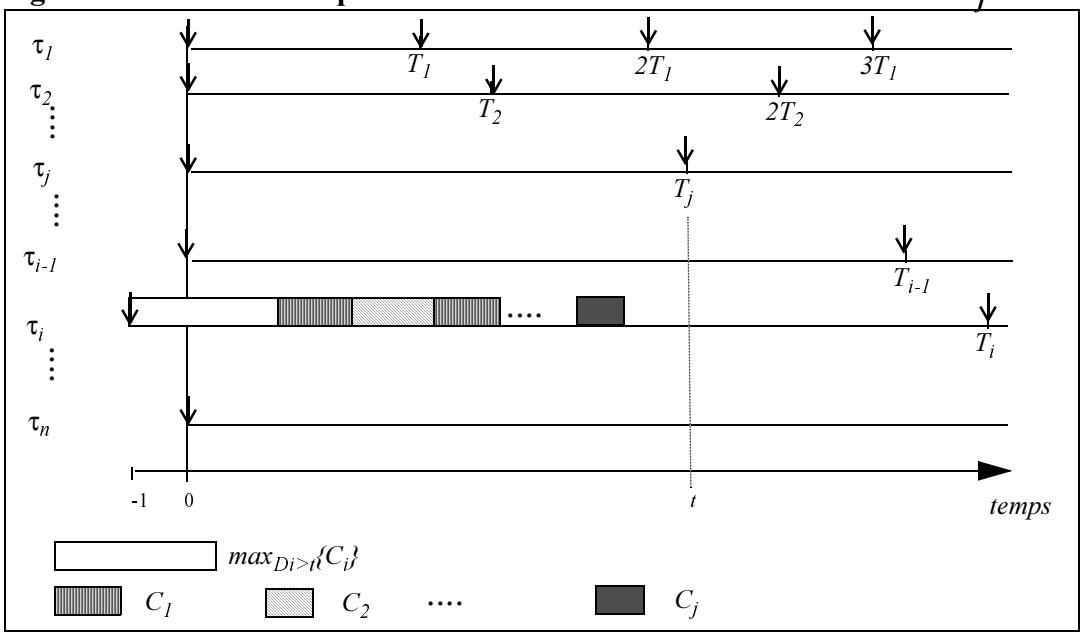
2. Les référentiels d'ordonnancement utilisés dans cette partie sont cependant moins généraux que ceux de la partie préemptive car nous ne pouvons plus utiliser la propriété d'équivalence en périodes et en échéances (cf. Définition 36 p. 52) sans durcir les conditions de faisabilité. Ce point sera précisé lors du chapitre IX.1. p. 85.

(27), c.à.d. à une contradiction. Nous renvoyons le lecteur au Lemme 17 p. 73 pour une généralisation et un raffinement de ce résultat, grâce au concept de *deadline busy period*.

Notons que dans le cas $D_i = T_i,\ \forall i \in [1,n]$, s'il existe $t > max\{D_i\}$ tel que $t < h(t) + max_{D_i > t}\{C_i - 1\} \leq \sum_{i=1}^{n}((t + T_i - D_i)/T_i)C_i \leq tU,$ alors $1 < U$. Le trafic ne peut donc avoir été déclaré faisable par l'éq. (27).

Une conséquence remarquable de ceci est que $\forall t > max\{D_i\}$, *EDF* et *NP-EDF* sont strictement équivalents puisqu'alors $max_{D_i > t}\{C_i - 1\} = 0$. De plus ce test est pseudo-polynomial puisque l'on teste un prédicat en *O(n)* sur un intervalle borné par le $max\{D_i\}$.

Figure 12 : Inversion de priorité maximale avec *NP-EDF* à l'instant $t=T_j$



[Shin and Zheng 1994] étendent cette propriété au cas des trafics généraux. Rappelons qu'il peut alors y avoir plus d'une activation par tâche en file d'attente sans pour autant rater d'échéances (cf. chapitre IV.2.3.4. p. 32). Leur modèle étant orienté vers la communication de messages, ils majorent cependant les inversions de priorités possibles.

Théorème 21 - *[Shin and Zheng 1994]. Soit un référentiel d'ordonnancement* $\Sigma=(\Upsilon,\Pi)$ *ou* Υ *contient tout TPN,* Π *contient NP-EDF et* C_p *est la durée maximale de tout message.* $\forall(\tau \in \Upsilon)$, τ *est NP-EDF-faisable si* $U < 1$ *et* $\forall t \in S,\ t \geq h(t) + C_p - 1,$

$$avec\ S = \left(\bigcup_{i=1}^{n}\{D_i + kT_i, k \in \mathbb{N}\}\right) \cap \left[0, max\left(max\{D_i\}, \frac{C_p - 1 + \sum_{i=1}^{n}(1 - D_i/T_i)C_l}{1 - U}\right)\right].$$

L'approche est identique au Théorème 4 p. 32 du cas préemptif : limitation du test à l'ensemble *S* des points de discontinuité ou $h(t)$ change de valeur ; prise en compte des bornes $max\{D_i\}$ pour les trafics généraux, et $(C_p - 1 + \sum_{i=1}^{n}(1 - D_i/T_i)C_l)/(1 - U)$ par manipulation algébrique de la demande processeur.

La différence avec le cas préemptif vient de la prise en compte des inversions de priorités qu'ils majorent systématiquement par $C_p - 1$, la pire durée de transmission d'un message. Ceci pose problème car, comme nous l'avons vu dans le cas $D_i = T_i$, $\forall i \in [1,n]$, les inversions de priorités disparaissent avec *NP-EDF* pour tout $t > max\{D_i\}$. La condition qu'ils obtiennent ne peut donc être nécessaire pour nos trafics non-concrets (cf. Hypothèse 1 p. 9).

VII.3. Priorités fixes

Les algorithmes d'ordonnancement à priorités fixes sont caractérisés par une assignation de priorités hors-ligne, valable pour toutes les activations d'une même tâche. Aucun résultat d'optimalité n'est connu dans le cas non-préemptif. Par contre, dans un modèle continu (c.à.d. où les paramètres des tâches, ainsi que les décisions d'ordonnancement, ne sont pas limités par des valeurs entières), [Hansson et al. 1994] et [Burns et al. 1995] adaptent le calcul des pires temps de réponse des tâches d'un trafic non-concret τ en contexte non-préemptif, non-oisif.

Théorème 22 - *[Hansson et al. 1994], [Burns et al. 1995]. Soit un référentiel d'ordonnancement* $\Sigma=(\Upsilon,\Pi)$ *ou* Υ *contient tout TPN et* Π *contient NP-PF, un algorithme d'ordonnancement non-préemptif, non-oisif à priorités fixes.* $\forall \tau \in \Upsilon$ *le pire temps de réponse* r_i *de toute tâche* τ_i *par NP-PF est obtenu par l'équation récursive suivante (où hp(i) dénote l'ensemble des tâches de plus haute priorité que* τ_i *et lp(i) l'ensemble des tâches de plus basse priorité que* τ_i*) :*

$$r_i = max_{q = 1\ldots Q}\{w_{i,q} + C_i - qT_i\}, \quad (28)$$

où Q est la valeur minimale telle que $w_{i,Q} + C_i \leq (Q+1)T_i$, Υ_{res} *est le temps de résolution et*

$$w_{i,q} = qC_i + \sum_{j \in hp(i)} \lceil (w_{i,q} + \Upsilon_{res})/T_j \rceil C_j + B_i, \text{ avec } B_i = max_{j \in lp(i)}\{C_j\}. \quad (29)$$

Ce résultat adapte en fait le Théorème 9 p. 39, établi dans un contexte préemptif. En dehors du paramètre Υ_{res}, introduit pour permettre d'aligner éventuellement les calculs sur le temps de résolution du système, les deux différences sont :

- le terme $B_i = max_{j \in lp(i)}\{C_j\}$ introduit dans chaque fenêtre $w_{i,q}$ (cf. éq. (29)) qui traduit l'inversion de priorité maximale provoquée par l'absence de préemption avec *NP-PF*. Remarquons que contrairement à *NP-EDF*, ce terme est une constante qui représente par tâche τ_i la plus grande durée d'exécution parmi les tâches ayant une priorité inférieure (c.à.d. parmi $lp(i)$).
- le déplacement de la durée d'exécution de la $(q+1)^{ième}$ activation de la tâche τ_i de l'éq. (29) vers l'éq. (28). Ceci traduit l'impossibilité de préempter cette activation dès lors qu'elle a commencé de s'exécuter. Le calcul de $w_{i,q}$ s'arrête alors sur le démarrage de la $(q+1)^{ième}$ activation de la tâche τ_i et non, comme en préemptif, sur sa fin d'exécution.

Une condition nécessaire et suffisante évidente pour tester la faisabilité du trafic non-concret τ revient alors à tester : $\forall i \in [1, n]$, $r_i \leq D_i$ (cf. Propriété 2 p. 11).

VIII. Extensions/discussions

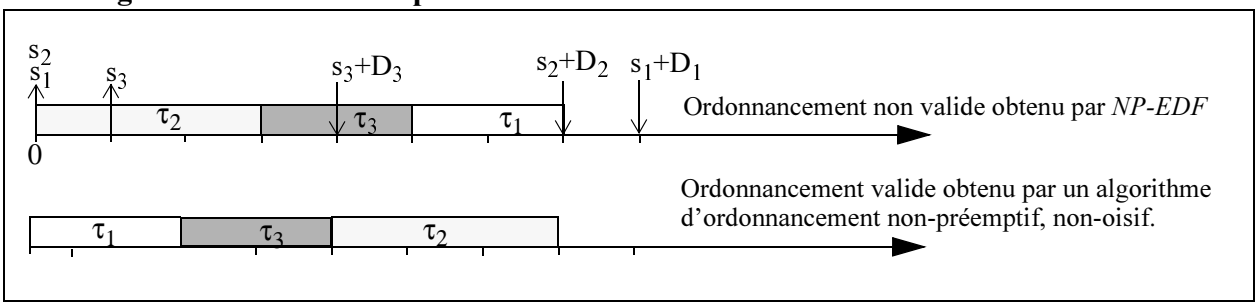
L'état de l'art est donc moins complet qu'en préemptif. *NP-EDF* semble être optimal parmi les algorithmes non-préemptifs non-oisifs (au moins dans le cas de trafics non concrets vérifiant $D_i = T_i, \forall i \in [1, n]$). Par contre, nous ne savons rien de l'optimalité dans le cas de priorités fixes. La formulation des tests se présente toujours en $\forall t \in S, P(t)$ ou en $\forall i \in [1, n], r_i \leq D_i$ mais nous ne savons pas comment calculer les pires temps de réponse avec *EDF* ou borner les intervalles d'étude par tâche. Nous revenons donc sur ces points en proposant quelques extensions et discussions en présence de trafics non-concrets. Nous insisterons sur les différences/ressemblances avec le cas préemptif. L'efficacité des algorithmes et le coût des tests associés seront examinés au chapitre IX. p. 85.

VIII.1. *Earliest Deadline First* Non-Préemptif, Non-Oisif (*NP-EDF*)

VIII.1.1. Une optimalité limitée

Observons tout de suite par un simple contre-exemple que contrairement au cas préemptif, où [Dertouzos 1974] établit l'optimalité de *EDF* par un argument de permutation très général (cf. Théorème 1 p. 29), *NP-EDF* n'est pas optimal en présence de trafics concrets ω parmi les algorithmes d'ordonnancement non-préemptifs, non-oisifs.

Figure 13 : *NP-EDF* en présence de trafic concrets



Par contre, en présence de trafics non-concrets τ, *NP-EDF* peut être montré Σ-optimal si l'on se limite aux algorithmes d'ordonnancement non-préemptifs, non-oisifs. Les référentiels d'ordonnancement concernés par cette propriété sont donc limités mais restent intéressants puisque *NP-EDF* sera alors capable d'ordonnancer tout scénario d'activation ω de τ, dès lors qu'un autre algorithme non-préemptif, non-oisif dans Σ sera capable de les ordonnancer (cf. chapitre III.3. p. 20 pour une discussion de la Σ-faisabilité et la Σ-optimalité en présence de trafics non-concrets).

Pour établir ce résultat d'optimalité, commençons par rappeler une condition de faisabilité **nécessaire** triviale pour tout algorithme d'ordonnancement non-préemptif oisif ou non-oisif en présence de trafic non-concrets τ. Nous focaliserons ensuite sur le concept de *deadline busy period* pour montrer que cette condition est aussi suffisante pour *NP-EDF* (cf. Théorème 23 p. 77). [Kim and Naghibdadeh 1980], puis plus formellement [Jeffay et al. 1991], établissent ce résultat lorsque le trafic non-concret vérifie $D_i = T_i, \forall i \in [1, n]$ (cf. chapitre VII.2. p. 69). [Shin and Zheng 1994] généralisent ce résultat mais pour un modèle de trafic différent.

Lemme 16 - *Si un trafic non-concret τ est faisable par un algorithme non-préemptif NP-P, alors :*

$$\forall t \geq 0, \quad t \geq \sum_{D_i \leq t} (1 + \lfloor (t - D_i)/T_i \rfloor) C_i + max_{D_i > t}\{C_i - 1\}$$

$$où\ max_{D_i > t}\{C_i - 1\} = 0\ si\ \nexists i: D_i > t$$

Preuve. Construisons ω, un scénario d'activation possible de τ. Soit 0, un instant synchrone pour toute tâche τ_i de τ vérifiant $D_i \leq t$. Comme l'inter-arrivée minimale de τ_i est T_i, il peut donc y avoir $1 + \lfloor (t - D_i)/T_i \rfloor$ activations de τ_i arrivées, et ayant une échéance absolue, dans l'intervalle $[0, t]$. En contexte non-préemptif non-oisif, la tâche vérifiant $max_{D_i > t}\{C_i - 1\}$ peut retarder l'exécution des tâches précédentes si elle est activée en -1 (ce retard s'annule si aucune tâche ne vérifie $D_i > t$). Ainsi pour cette instanciation concrète ω de τ, et quelque soit t positif, il doit être possible d'exécuter $\sum_{D_i \leq t}(1 + \lfloor (t - D_i)/T_i \rfloor)C_i + max_{D_i > t}\{C_i - 1\}$ dans l'intervalle $[0, t]$ si par hypothèse τ est faisable par *NP-P*. ❐

VIII.1.2. Deadline-d busy period

Nous adaptons ici les résultats du chapitre V.1.1. p. 41 au contexte non-préemptif, non-oisif[1]. Le Lemme 2 p. 17, nous prouvant que L est la plus longue période occupée, nous focalisons à nouveau sur les *deadline-d busy period* pour en déduire $L_i \leq L$, la plus grande *deadline busy period* pour toute tâche τ_i de τ, afin de borner l'intervalle d'étude permettant d'établir la faisabilité d'un trafic ou de calculer les pires temps de réponse avec *NP-EDF*.

VIII.1.2.1. Propriétés

Rappelons qu'une *deadline-d busy period* est caractérisée par un intervalle où seules des activations de tâches ayant une échéance absolue inférieure ou égale à d sont exécutées (cf. Définition 19 p. 19). La différence avec le cas préemptif est ici qu'une inversion de priorité peut se produire avant toute *deadline-d busy period*, provoquée par une activation de tâche ayant une échéance supérieure à d. Pour la faisabilité seule, nous adaptons le Lemme 9 p. 41, en le faisant précéder de la pire inversion de priorité possible.

Lemme 17 - *[George et al. 1996]. Soit un trafic non-concret τ ordonnancé par NP-EDF. Si une échéance absolue est ratée pour un certain scénario d'activation alors $\exists t \geq 0$ tel que h(t)+B(t)>t dans une deadline-t busy period résultant du scénario d'activation suivant. τ_j vérifiant $B(t) = max_{D_j > t}\{C_j - 1\}$ est activée à l'instant -1 ($B(t) = 0$ si $\forall j, \quad D_j \leq t$) et toutes les autres tâches du trafic sont synchrones en 0 et périodiques ensuite.*

Preuve. Soit un scénario d'activation ω de τ conduisant une activation de la tâche τ_i à rater une échéance absolue à l'instant t_2. Soit t_1 le dernier instant avant t_2 tel qu'il n'y ait pas d'activation de tâche en file d'attente ayant une échéance absolue inférieure ou égale à t_2 (nous posons $t_1 = 0$ *et* $t_2 = t$ dorénavant, cf. Figure 14). Par choix :

- *0* doit être l'instant d'activation d'une tâche et il n'y a pas de période inoccupée du processeur dans l'intervalle $[0, t]$ puisque *NP-EDF* est non-oisif.
- une inversion de priorité peut se produise en *0* puisque *NP-EDF* est non-préemptif.

Soit b la durée d'une telle inversion de priorité. Avec *NP-EDF,* b est provoquée par une tâche ayant une échéance relative supérieure à t et seules des activations de tâches ayant des échéances absolues inférieures ou égales à t sont exécutées dans l'intervalle $[b, t]$. L'instant b est donc le début d'une *deadline-t busy period*, dont la durée est supérieure à $t - b$, par l'hypothèse qu'une échéance absolue est ratée en t.

Soit τ_j la tâche vérifiant $B(t) = max_{D_j > t}\{C_j - 1\}$. $B(t)$ est donc la durée maximale d'une inversion de priorité avec *NP-EDF* si τ_j démarre son exécution à l'instant -1. En effet toute autre inversion de priorité a une durée $b \leq B(t)$ (notons que $B(t) = 0$ si $\forall j, \quad D_j \leq t$).

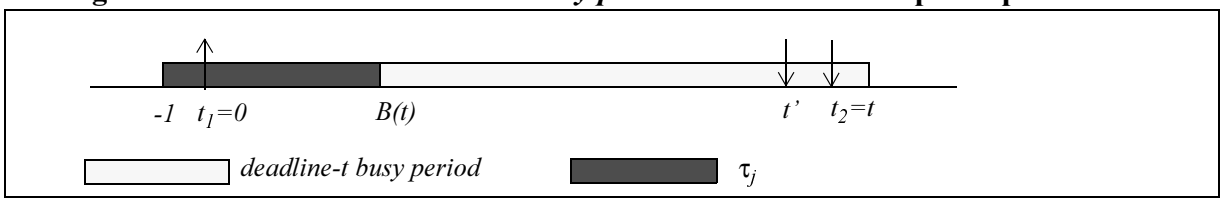
Considérons maintenant le scénario périodique pour lequel τ_j est activé en -1 et où toutes les autres tâches, sauf τ_i, sont synchrones en *0* (notons que toutes les tâches, sauf τ_i, sont

1. ces résultats ont été établis avec notre co-auteur M. Spuri dans [George et al. 1996].

synchrones en *0* si $B(t) = 0$). Comme nous provoquons la pire inversion de priorité, le début de la *deadline-t busy period* est repoussé en $B(t)$. De plus, la demande processeur ne peut diminuer dans l'intervalle $[0, t]$ car le nombre d'activations de tâches ayant une échéance absolue dans cet intervalle ne peut qu'augmenter. La longueur de la *deadline-t busy period* démarrant en $B(t)$ ne peut donc diminuer et τ_i rate toujours une échéance à l'instant t.

Finalement si τ_i est aussi activée de façon synchrone en *0* et périodique ensuite, alors on a le scénario d'activation énoncé par ce lemme et la demande processeur dans l'intervalle $[0, t]$ est maximisée. Elle vaut maintenant *h(t)* (cf. Définition 15 p. 16) qui donne la durée maximale de la *deadline-t busy period* démarrant en $B(t)$. Soit t' la plus grande échéance absolue précédant t dans ce dernier scénario, on a *h(t)=h(t')*. Comme *B(t)+h(t)* est plus grand que $t \geq t'$, alors une échéance absolue est ratée au plus tard en t' dans la *deadline-t busy period* démarrant en *B(t)*. ❒

Figure 14 : Faisabilité et *Deadline busy period* en contexte non-préemptif non-oisif



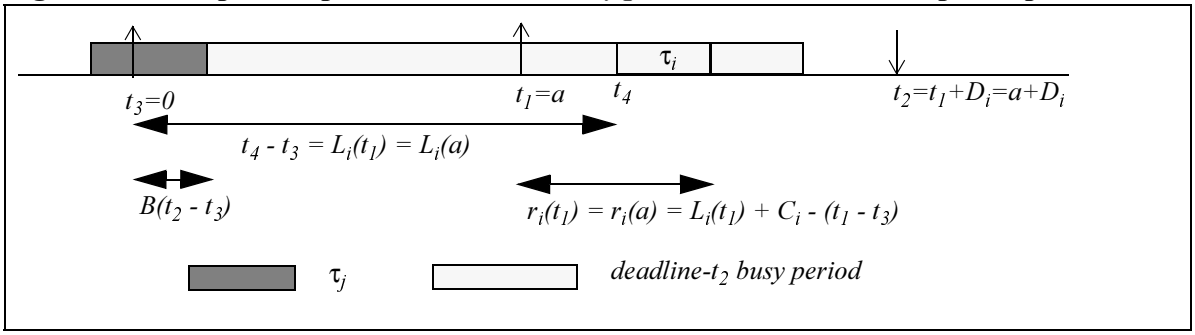
De la même façon, le concept de *deadline busy period* permet de limiter les intervalles où s'effectue le calcul des pires temps de réponse en contexte non-préemptif, non-oisif.

Lemme 18 - *[George et al. 1996]. Avec EDF, le pire temps de réponse d'une tâche τ_i de τ se trouve dans une deadline($a + D_i$) busy period résultant d'une activation périodique des tâches où : τ_i est activée en a ; toutes les autres tâches ayant une échéance absolue inférieure ou égale à $a + D_i$ sont activées de façon synchrone en 0 ; la tâche (s'il en existe une) ayant la plus grande durée d'exécution, parmi celles ayant une échéance absolue supérieure à $a + D_i$, est activée à l'instant -1.*

Preuve. Soit un scénario ω de τ où τ_i est activée à l'instant t_1 avec son échéance absolue en $t_2 = t_1 + D_i$ (cf. Figure 15). Soit t_4, l'instant de début d'exécution par *NP-EDF* de cette activation et t_3, le dernier instant avant t_1 tel qu'il n'y ait pas d'activation de tâche en file d'attente ayant une échéance absolue inférieure ou égale à t_2. Par choix :

- t_3 est l'instant d'activation d'une tâche et il ne peut y avoir de période inoccupée dans l'intervalle $[t_3, t_4]$ puisque *NP-EDF* est non-oisif.
- l'exécution de l'activation de τ_i arrivée en t_1 est précédée d'une période occupée constituée d'activations de tâches arrivées dans l'intervalle $[t_3, t_4]$ et dont les échéances absolues sont inférieures ou égales à t_2. Cette période est donc une *deadline-t_2 busy period*.
- une activation de tâche précédant t_3 et ayant une échéance absolue supérieure à t_2 peut créer une inversion de priorité avant cette *deadline-t_2 busy period*, car *NP-EDF* est non-préemptif.

Figure 15 : Temps de réponse et *Deadline busy period* en contexte non-préemptif non-oisif



Considérons maintenant le scénario pour lequel:

- toutes les tâches, sauf τ_i, ayant des échéances relatives inférieures ou égales à $t_2 - t_3$ sont activées de façon synchrone en *0* et périodique ensuite,
- τ_i est activée en ..., *a-2T*$_i$, *a-T*$_i$, *a*, *a+T*$_i$..., avec $a = t_1 - t_3$,
- τ_j tel que $B(t_2 - t_3) = \max_{D_j > t_2 - t_3}\{C_j - 1\}$ (s'il existe une telle tâche) est activée en *-1*.

τ_j provoque la pire inversion de priorité par *NP-EDF* pour l'échéance absolue $a + D_i$ et la demande processeur induite est maximisée dans $[0, a + D_i]$, par rapport à celle induite dans $[t_3, t_2]$ initialement. Le début d'exécution de l'activation de τ_i arrivée en a est donc retardé par rapport à l'instant t_4 dans le scénario initial et donc au-delà de a dans le nouveau scénario. En appliquant ce raisonnement quelque soit le scénario initial, on obtient nécessairement le pire temps de réponse de τ_i sur l'un d'entre eux. ❒

VIII.1.2.2. Calcul de L_i

Comme en préemptif, pour toute tâche τ_i il est possible de calculer en non-préemptif non-oisif la durée de L_i (cf. Définition 20 p. 20), la plus longue *deadline busy period* possible pour τ_i. Ceci, grâce au dernier lemme, va nous permettre de restreindre à $[0, L_i]$, l'intervalle où est calculé le pire temps de réponse de τ_i possible par *NP-EDF*. Le calcul s'inspire de celui proposé au chapitre IV.2.4. p. 33, pour le cas préemptif, mais présente deux différences :

- (D1) le terme $B(t) = max_{D_i > t}\{C_i - 1\}$ traduit l'inversion de priorité maximale provoquée par *NP-EDF* à l'instant *t*. Nous avons déjà remarqué que $B(t)$ s'annule pour tout $t > max\{D_i\}$. *EDF* et *NP-EDF* sont donc similaires après $max\{D_i\}$.
- (D2) toujours en raison de l'absence de préemption, le calcul de $L_i(a)$ proposé en contexte préemptif doit être légèrement modifié pour s'arrêter sur le début d'exécution de l'activation de tâche arrivée en *a*, et non pas sur sa fin d'exécution. Ceci car l'absence de préemption interdit d'interrompre l'exécution de cette activation.

Plus précisément, pour un des scénario d'activation ω du Lemme 18 p. 74, si la durée de la période occupée démarrant en *0* et précédant l'exécution de l'activation de τ_i arrivée en a, est appelé $L_i(a)$ (cf. Figure 15, avec t_3=*0* et t_1=*a*), le temps de réponse de cette activation vaut alors $r_i(a) = L_i(a) + C_i - a$ (et non $L_i(a) - a$ comme en préemptif). En adaptant le calcul du chapitre IV.2.4. p. 33, pour tenir compte du contexte non-préemptif non oisif, la durée de $L_i(a)$ est alors obtenue par la première racine de l'équation récursive suivante :

$$L_i^{(m+1)}(a) = B(a + D_i) + \overline{W}_i(a, L_i^{(m)}(a)) + \lfloor a/T_i \rfloor C_i. \qquad (30)$$

Le premier terme du côté droit de l'équation, $B(a + D_i) = max_{D_j > a + D_i}\{C_j - 1\}$, tient compte de l'inversion de priorité maximale par *NP-EDF* à l'instant $a + D_i$. Le second terme, $W_i(a,t)$, calcule la durée d'exécution maximal d'activations de tâches (autres que τ_i) ayant des échéances absolues inférieures ou égales à $a + D_i$ et étant arrivées durant $L_i(a)$. Plus précisément pour tout *t* et pour toute tâche τ_j, le nombre maximum d'activations de τ_j arrivées dans l'intervalle fermé $[0,t]$ est égal à $1 + \lfloor t/T_j \rfloor$. Contrairement au cas préemptif, nous utilisons un intervalle fermé à droite car $L_i(a)$ ne s'arrête plus sur la fin d'exécution de l'activation de τ_i arrivée en a mais sur son début d'exécution. Il ne doit donc pas y avoir d'activation de tâche plus prioritaire à cet instant. Par contre, comme dans le cas préemptif nous bornons le nombre d'activations de tâches pertinentes par $1 + \lfloor (a + D_i - D_j)/T_j \rfloor$, c.à.d. par celles ayant une échéance absolue inférieures ou égales à $a + D_i$. Nous obtenons donc:

$$\overline{W}_i(a, t) = \sum_{j \neq i,\, D_j \leq a + D_i} min\{1 + \lfloor t/T_j \rfloor, 1 + \lfloor (a + D_i - D_j)/T_j \rfloor\} C_j. \qquad (31)$$

Le troisième terme enfin donne le temps nécessaire à l'exécution des activations de τ_i arrivées avant a durant la *deadline-*$a + D_i$ *busy period.*

Pour tout $a \geq 0$, la récursion s'arrête dès que $L_i^{(m)}(a) = L_i^{(m+1)}(a)$, $L_i(a)$ vaut alors $L_i^{(m)}(a)$. L'éq. (30) est convergente si $U \leq 1$ car l'éq. (31) (et donc l'éq. (30)), sont croissantes en *a,* et car $L_i(a) \leq L$ et L est bornée si $U \leq 1$ (cf. Lemme 2 p. 17, Lemme 3 p. 18).

En nous basant sur les scénarii d'activation ω du Lemme 18 p. 74, nous savons donc calculer $L_i(a)$ dans le cas non-préemptif non-oisif. Il est possible d'en déduire la borne L_i, utilisée pour le calcul du pire temps de réponse de la tâche τ_i, en testant toutes les valeurs possibles de *a*. Il est aussi possible de réutiliser la procédure hors-ligne établie dans le cas préemptif (cf. chapitre V.1.1.2. p. 43) en remplaçant toutefois la formule de $L_i(a)$ par celle qui vient d'être établie ici. Ces résultats s'appliquent aussi pour la borne L_i^s utilisée pour l'analyse de la faisabilité seule, avec $L_i^s \leq L_i \leq L$ puisque l'on considère alors uniquement le scénario du Lemme 17 p. 73, qui est inclu dans ceux du Lemme 18 p. 74.

VIII.1.3. Optimalité et faisabilité

Le prédicat utilisé comprend d'une part la demande processeur $h(t)$ (cf. Définition 15 p. 16) pour toutes les tâches ayant une échéance relative inférieures ou égale à *t* et d'autre part $B(t) = max_{D_i > t}\{C_i - 1\}$, la pire inversion de priorité en *t* ($B(t) = 0$ si $\nexists i$: $D_i > t$, puisqu'il ne peut alors y avoir d'inversion de priorité).

$$h(t) + B(t) = \sum_{i=1}^{n} max(0, 1 + \lfloor (t - D_i)/T_i \rfloor) C_i + max_{D_i > t}\{C_i - 1\}. \qquad (32)$$

En s'inspirant des résultats préemptifs deux approches permettent de borner l'intervalle d'étude, manipulations algébriques de la demande processeur et périodes occupées. Nous adaptons trivialement le Lemme 13 p. 45, sur la borne *L*, pour établir la faisabilité avec *NP-EDF* en tenant compte des inversions de priorités maximales.

Lemme 19 - $\forall \boldsymbol{t}, \boldsymbol{0} \leq \boldsymbol{t} \leq L, \Phi(\boldsymbol{t}) \geq \boldsymbol{0} \Rightarrow \forall \boldsymbol{t}, \boldsymbol{t} \geq \boldsymbol{0}, \Phi(\boldsymbol{t}) \geq \boldsymbol{0}$ *avec* $\Phi(t) = t - (h(t) + B(t))$.

Preuve. Soit $\Delta_L(t) = \Phi(t + L) - \Phi(t)$, nous avons :

$$\Delta_L(t) = L - \sum_{j=1}^{n} \left(max\left\{0, 1 + \left\lfloor \frac{t + L - D_j}{T_j} \right\rfloor\right\} - max\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} \right) C_j$$

$$-(max_{D_i > t + L}\{C_i - 1\} - max_{D_i > t}\{C_i - 1\})$$

$$\geq L - \sum_{j=1}^{n} \left(max\left\{0, 1 + \left\lfloor \frac{t + L - D_j}{T_j} \right\rfloor\right\} - max\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\} \right) C_j$$

$$\geq L - \sum_{j=1}^{n} max\left\{0, \left\lfloor \frac{L}{T_j} + \frac{t - D_j}{T_j} - \left\lfloor \frac{t - D_j}{T_j} \right\rfloor \right\rfloor\right\} C_j \geq L - \sum_{j=1}^{n} \left\lceil \frac{L}{T_j} \right\rceil C_j$$

Comme L est la première racine de $L = \sum_{j=1}^{n} \lceil L/T_j \rceil C_j$ (cf. chapitre III.2.1. p. 17), $\Delta_L(t) = \Phi(t + L) - \Phi(t) \geq 0$. Si $\forall t, 0 \leq t \leq L, \Phi(t) \geq 0$, le résultat est alors prouvé. ❐

En adaptant l'état de l'art à nos trafic non-concrets τ et en intégrant les bornes L_i et L, nous obtenons un résultat d'optimalité pour *NP-EDF* et un test nécéssaire et suffisant pour établir la faisabilité de τ par cet algorithme.

Théorème 23 - *[George et al. 1996]. Soit un référentiel d'ordonnancement* $\Sigma=(\Upsilon,\Pi)$ *où* Υ *contient tout TPN et* Π *contient tout algorithme d'ordonnancement non-préemptif non-oisif. NP-EDF est* Σ*-optimal et* $\forall(\tau \in \Upsilon)$, τ *est NP-EDF-faisable si et seulement si :*

$$\forall t \in S,\ \sum_{i=1}^{n} max(0, 1 + \lfloor (t - D_i)/T_i \rfloor) C_i + max_{D_i > t}\{C_i - 1\} \leq t$$

$$\text{avec } max_{D_i > t}\{C_i - 1\} = 0 \text{ lorsque } \nexists i: D_i > t,$$

$$S = \left(\bigcup_{i=1}^{n} \left\{ kT_i + D_i, 0 \leq k \leq \left\lceil (min\{\mu, L, L_i^s\} - D_i)/T_i \right\rceil + 1 \right\} \right),$$

$$\mu = max\left\{ max\{D_j\}, \sum_{j=1}^{n} (T_j - D_j)U/(1-U) \right\},$$

$$L = \sum_{j=1}^{n} \lceil L/T_j \rceil C_j \text{ (la première racine de cette équation),}$$

L_i^s *(la plus grande deadline busy period synchrone pour la tâche* τ_i, $\forall i \in [1, n]$*).*

Preuve. $h(t)$ mesure la demande processeur maximale dans l'intervalle $[0, t]$ à partir du scénario d'activation synchrone en *0* (cf. Définition 15 p. 16). Cette condition est nécessaire pour tout algorithme non-préemptif en présence de trafics non-concrets τ (cf. Lemme 16 p. 72). Elle est aussi suffisante pour *NP-EDF* car si un scénario d'activation rate une échéance absolue par *NP-EDF*, alors il existe un instant *t* sur un scénario du Lemme 17 p. 73, conduisant à vérifier que *h(t)+B(t)>t*. *NP-EDF* est donc Σ-optimal parmi les algorithmes non-préemptifs non-oisifs pour les trafics non-concrets τ.

$h(t) + B(t)$ étant une fonction discontinue, $(h(t) + B(t))/t$ décroît entre deux points de discontinuité. Seuls les points dans l'ensemble *S* sont donc pertinents.

une deuxième conséquence du Lemme 17 p. 73, est la borne L_i^s sur l'intervalle d'étude (cf. chapitre VIII.1.2.2. p. 75). La borne L vient du Lemme 19 p. 76. Notons que $L_i^s \leq L$, mais le calcul proposé pour L_i^s étant procédural, nous conservons la borne L dans le test.

Enfin, la borne μ adapte le résultat du Théorème 21 p. 70, qui majore les inversions de priorité par la constante C_p. Si le test est vérifiée dans l'intervalle *[0,*μ*]* mais rate une échéance en *t*>μ, alors comme par hypothèse $\mu \geq Max\{D_j\}$, nous avons $B(t) = 0$ et donc :

$$t < h(t) + B(t) \leq \sum_{i=1}^{n} ((t + T_i - D_i)/T_i) C_i \leq tU + \sum_{i=1}^{n} ((T_j - D_j)/T_i) C_i,$$

$$\text{c.à.d. } t < \sum_{j=1}^{n} (T_j - D_j)U/(1-U) \text{ (contradiction).} \qquad \square$$

Ce test est pseudo-polynomial si $U \leq c < 1$ (cf. chapitre IX.2.1. p. 89). Du point de vue de la faisabilité, il n'est que suffisant pour *EDF* car il introduit un terme additionnel dans le prédicat à tester du Théorème 12 p. 46 (un test nécéssaire et suffisant pour *EDF*). Un trafic faisable par *NP-EDF* est donc faisable par *EDF* (nous verrons que cette remarque ne tient pas avec les algorithmes à priorités fixes).

Rappelons que la Σ-optimalité énoncée ici pour *NP-EDF* limite le référentiel d'ordonnancement Σ aux algorithmes non-préemptifs non-oisifs en présence de trafics non-concrets τ. Elle est toutefois intéressante puisqu'elle couvre tout algorithme non-préemptif non-oisif pouvant ordonnancer tout scénario d'activation ω de τ (cf. chapitre III.3.2. p. 22 pour une discussion de la Σ-faisabilité et de la Σ-optimalité).

VIII.1.4. Faisabilité basée sur les pires temps de réponse

L'état de l'art ne traite pas cette approche. Le Lemme 18 p. 74 conduit à une *deadline($a+D_i$) busy period* telle que :

- toutes les tâches, sauf τ_i, ayant une échéance relative inférieure ou égale à $a+D_i$ sont activées de façon synchrones en *0* et périodiquement ensuite (c.à.d. $s_j = 0$, $\forall (j \neq i), D_j \leq a + D_i$),
- τ_i est activée en ..., $a-T_i$, a, $a+T_i$... (c.à.d. $s_i = a - \lfloor a/T_i \rfloor T_i$),
- la tâche ayant la plus grande durée d'exécution parmi celles ayant une échéance absolue supérieure à $a + D_i$ (s'il en existe une), est activée à l'instant $t = -1$.

Pour ce scénario d'activation, $r_i(a) = max\{C_i, L_i(a) + C_i - a\}$ où $L_i(a)$ est obtenu par l'équation (30) p. 75, que nous exprimons sous la forme non condensée :

$$L_i(a) = max_{D_j > a + D_i}\{C_j - 1\} + \sum_{j \neq i,\ D_j \leq a + D_i} min\left\{1 + \left\lfloor \frac{L_i(a)}{T_j} \right\rfloor, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor\right\} C_j + \left\lfloor \frac{a}{T_i} \right\rfloor C_i. \qquad (33)$$

Adaptons le Lemme 14 p. 47, au cas non-préemptif non-oisif afin d'établir la borne *L*, sur les valeurs de *a* pertinentes, pour le calcul des pires temps de réponse avec *NP-EDF*.

Lemme 20 - $\forall a,\ a \geq 0,\ r_i(a + L) \leq r_i(a)$

Preuve. Si par hypothèse $L_i^k(a + L) - L_i^k(a) \leq L$, alors :

$$\begin{aligned} L_i^{(k+1)}(a + L) &= max_{D_j > a + L + D_i}\{C_j - 1\} + \lfloor (a + L)/T_i \rfloor C_i \\ &\quad + \sum_{j \neq i} min\{1 + \lfloor L_i^{(k+1)}(a + L)/T_j \rfloor, max\{0, 1 + \lfloor (a + L + D_i - D_j)/T_j \rfloor\}\} C_j \\ &\leq max_{D_j > a + D_i}\{C_j - 1\} + \lfloor a/T_i \rfloor C_i \\ &\quad + \sum_{j \neq i} min\{1 + \lfloor (L_i^{(k)}(a))/T_j \rfloor, max\{0, 1 + \lfloor (a + D_i - D_j)/T_j \rfloor\}\} C_j + \sum_j \lceil L/T_j \rceil C_j \\ &= L_i^{(k+1)}(a) + L \end{aligned}$$

Comme par définition $L_i^{(0)}(a + L) - L_i^{(0)}(a) \leq L$, alors par récurrence $L_i(a + L) - L_i(a) \leq L$. Comme de plus $r_i(a + L) - r_i(a) = L_i(a + L) - L_i(a) - L$, le résultat est prouvé. ❐

Nous obtenons le test nécéssaire et suffisant suivant pour établir la faisabilité de tout trafic non-concret τ par *NP-EDF* en se basant sur le calcul des pires temps de réponse.

Théorème 24 - *[George et al. 1996]. Soit un référentiel d'ordonnancement* $\Sigma=(\Upsilon,\Pi)$ *ou* Υ *contient tout TPN et* Π *contient NP-EDF.* $\forall(\tau \in \Upsilon)$, τ *est NP-EDF-faisable si et seulement si :*

$$\forall i \in [1, n], r_i \leq D_i \text{ où } r_i = max_{a \in S}\{C_i, L_i(a) + C_i - a\} \text{ avec:}$$

$$L_i(a) = max_{D_j > a + D_i}\{C_j - 1\} + \sum_{j \neq i,\ D_j \leq a + D_i} min\left\{1 + \left\lfloor \frac{L_i(a)}{T_j} \right\rfloor, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor\right\} C_j + \left\lfloor \frac{a}{T_i} \right\rfloor C_i,$$

$$S = (\bigcup_{j=1}^{n}\{kT_j + D_j - D_i, 0 \leq k \leq \lfloor min\{L, L_i\}/T_j \rfloor\}),$$

$$L = \sum_{j=1}^{n} \lceil L/T_j \rceil C_j \text{ (la première racine de cette équation),}$$

L_i *(la plus grande deadline busy period pour la tâche* τ_i, $\forall i \in [1, n]$*).*

Preuve. $L_i(a)$ donne la taille maximale d'une *deadline(a+D_i) busy period* correspondant à l'un des scénarii d'activation ω de τ du Lemme 18 p. 74. Cette condition est trivialement nécessaire pour *NP-EDF* car ces scénarii sont des instanciations concrètes possibles de τ et $\forall i \in [1, n], \forall a, r_i(a) = L_i(a) + C_i - a \leq D_i$ si par hypothèse τ est faisable par *NP-EDF*. Elle est suffisante pour *NP-EDF* car tout temps de réponse dans un scénario d'activation ne peut qu'augmenter en le ramenant au scénario correspondant du Lemme 18 p. 74. Les bornes L_i et L sur l'intervalle d'étude viennent respectivement du chapitre VIII.1.2.2. p. 75, et du Lemme 20 p. 78. $L_i \leq L$, mais le calcul de L_i étant procédural, nous conservons L dans le test. ❐

Ce test est pseudo-polynomial si $U \leq c < 1$ (cf. chapitre IX.2.3. p. 90). Notons que malgré les inversions de priorités, les pires temps de réponse par *NP-EDF* ne sont pas nécessairement supérieurs à ceux de *EDF* car le calcul de $L_i(a)$ est différent (cf. chapitre VIII.1.2.2. p. 75).

VIII.2. Priorités fixes

Nous précisons[1] ici l'utilisation des *level-i busy periods* dans le cas non-préemptifs puis discutons des problèmes posés par l'optimalité dans ce contexte où aucun résultat n'est connu. Rappelons toutefois que le Théorème 22 p. 71, de l'état de l'art énonce déja la faisabilité de tout trafic non-concret τ dans le cas non-préemptif. On notera $lp(i)$ l'ensemble des tâches de plus basse priorité que τ_i, et $hp(i)$ celui des tâches de plus haute priorité que τ_i.

VIII.2.1. Level-i busy period

VIII.2.1.1. Propriété

Il est possible d'adapter pour une tâche τ_i le concept de *level-i busy period*, introduit par Lehoczky en préemptif (cf. Lemme 8 p. 38), au contexte non-préemptif non-oisif. Rappelons que durant une telle période, seules des activations de tâches ayant une priorité supérieure ou égale à τ_i sont exécutées (cf. Définition 18 p. 19).

Lemme 21 - *[George et al. 1996]. Soit NP-PF un algorithme d'ordonnancement non-préemptif, non-oisif, à priorités fixes. Le pire temps de réponse de toute tâche τ_i d'un trafic non-concret τ se trouve dans une level-i busy period résultant du scénario d'activation strictement périodique suivant : toutes les tâches τ_j vérifiant $j \in hp(i) \cup \{i\}$ sont synchrones à l'instant 0 et la tâche τ_k vérifiant $B_i = max_{k \in lp(i)}\{C_k - 1\}$ est activée à l'instant -1 ($B_i = 0$ si $lp(i) = \emptyset$).*

Preuve. Soit l'ordonnancement produit par *NP-PF* sur un scénario d'activation ω de τ (cf. Figure 16). Soit t_2 le début d'exécution d'une des activations de τ_i et t_1 le dernier instant avant t_2 sans activation de tâche en file d'attente de priorité supérieure ou égale à τ_i. Par choix :

- t_1 est l'instant d'activation d'une tâche τ_j vérifiant $j \in hp(i) \cup \{i\}$. Comme de plus *NP-PF* est non-oisif il ne peut pas y avoir de période inoccupée dans l'intervalle $[t_1, t_2]$.
- t_2 est précèdé par une période occupée ou seules des activations de tâches appartenant à $hp(i) \cup \{i\}$ sont exécutées (il s'agit donc d'une *level-i busy period*).
- en raison de l'absence de préemption, juste avant cette *level-i busy period*, une activation de tâche arrivée avant t_1 et ayant une priorité inférieure à τ_i (c.à.d. appartenant à $lp(i)$) peut s'exécuter en provoquant une inversion de priorité. La fin de cette exécution (ou l'instant t_1, s'il n'y a pas d'inversion de priorité) et t_2 délimitent la *level-i busy period.*

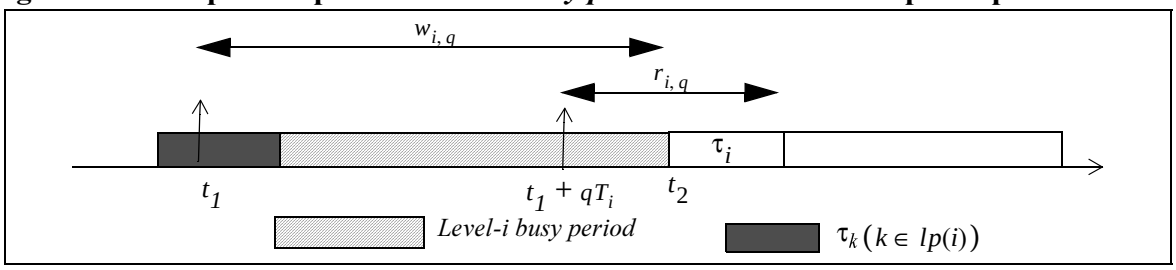
Si nous activons maintenant τ_i de façon synchrone en t_1 et périodiquement ensuite, les exécutions des activations de τ_i dans l'intervalle $[t_1, t_2]$ ne sont pas déplacées dans la *level-i busy period*. Par contre, les temps de réponse des activations de τ_i dans cet intervalle ne peuvent diminuer puisque les instants d'activation de τ_i sont décalés vers la gauche.

1. cette discussion est largement initiée par notre co-auteur L. George dans [George et al. 1996].

Si toutes les tâches τ_j de priorités supérieures à τ_i (c.à.d. $j \in hp(i)$) sont activées de façon synchrone en t_1 et périodiquement ensuite, leur nombre est maximisé et les temps de réponse des activations de τ_i dans cet intervalle ne peuvent qu'être inchangés ou augmentés.

Si enfin, parmi toutes les tâches τ_k de priorités inférieure à τ_i (c.à.d. $k \in lp(i)$), nous débutons l'exécution de celle vérifiant $B_i = max_{k \in lp(i)}\{C_k - 1\}$ en $t_1 - 1$ ($B_i = 0$ si $lp(i) = \emptyset$), alors l'effet de l'absence de préemption est maximisé et les temps de réponse des activations de τ_i dans cet intervalle ne peuvent qu'être inchangés ou augmentés. Finalement, en substituant t_1 par 0, le résultat est prouvé. ❒

Figure 16 : Temps de réponse et *level-i busy period* en contexte non-préemptif non-oisif



VIII.2.1.2. Calcul de L_i et de $w_{i,q}$

Grâce au lemme précédent, il est possible pour une tâche τ_i d'adapter les calculs proposés en contexte préemptif de L_i et de $w_{i,q}$ dans le cas non-préemptif, non-oisif. Pour cela, nous devons prendre en compte deux différences :

- le terme $B_i = max_{k \in lp(i)}\{C_k - 1\}$ qui traduit les inversions de priorités provoquées par l'absence de préemption (avec $B_i = 0$ si $lp(i) = \emptyset$). Notons que le blocage résultant est une constante qui représente pour τ_i la plus grande durée d'exécution parmi les tâches ayant une priorité inférieures (c.à.d. parmi $lp(i)$). Il impacte donc toute la *level-i busy period* et ne disparaît pas après $t > max\{D_i\}$ comme c'est le cas avec *NP-EDF*.
- pour $w_{i,q}$, le calcul porte sur l'intervalle fermé précédant l'exécution de la $(q+1)^{ième}$ activation de τ_i et non pas, comme en préemptif, sur l'intervalle ouvert terminant sur sa fin d'exécution (cf. Théorème 9 p. 39). En effet, en non-préemptif tant que l'exécution de la $(q+1)^{ième}$ activation de τ_i n'a pas démarré, il est possible qu'une activation de tâche plus prioritaire la retarde. Par contre, il n'est plus possible de la préempter ensuite.

Pour L_i (la plus longue *level-i busy period*), le calcul tient compte du terme B_i et, comme en préemptif, de la charge cumulée des activations des tâches de priorités supérieures ou égales à τ_i durant $[0, L_i[$ (c.à.d. $W_i(L_i) = \sum_{j \in hp(i) \cup \{i\}} \lceil L_i / T_j \rceil C_j$, cf. Définition 14 p. 15). Plus précisément pour l'intervalle $[0, t[$ ouvert à droite, car on ne veut pas prendre en compte d'activations arrivant en fin de L_i faisant parties d'une nouvelle *level-i busy period*, l'idée est d'adapter le calcul de L (cf. chapitre III.2.1. p. 17), en comparant l'addition de ces deux termes avec la longueur t de l'intervalle. Si le résultat est plus grand que t, alors L_i est nécessairement supérieur et l'argument est reconduit récursivement jusqu'à ce qu'on trouve une valeur égale à la précédente. Formellement, L_i est le point fixe de l'équation récursive convegente suivante :

$$\begin{cases} L_i^{(1)} = B_i + \sum_{j \in hp(i) \cup \{i\}} C_j \\ L_i^{(m+1)} = B_i + W_i(L_i^{(m)}) \end{cases} \quad (34)$$

Le calcul est arrêté lorsque $L_i^{(m)} = L_i^{(m+1)}$. L_i vaut alors $L_i^{(m)}$. De cette façon, nous trouvons la plus petite solution de l'équation (avec $max_{k \in lp(i)}\{C_k - 1\} = 0$ si $lp(i) = \emptyset$) :

$$L_i = max_{k \in lp(i)}\{C_k - 1\} + \sum_{j \in hp(i) \cup \{i\}} \lceil L_i / T_j \rceil C_j. \quad (35)$$

Pour $w_{i,q}$, rappelons qu'en non-préemptif non-oisif le calcul doit porter sur l'intervalle fermé précédant l'exécution de la *(q+1)ième* activation de τ_i (cf. Figure 16). Il doit donc prendre en compte le terme B_i, q activations de τ_i (la *(q+1)ième* est exclue car par hypothèse elle ne peut être préemptée) et $\overline{W_i}(t)$, la charge cumulée sur un intervalle fermé $[0, t]$ (cf. Définition 14 p. 15) du nombre maximum d'activations des tâches τ_j de priorités supérieures à τ_i, c.à.d. :

$$\overline{W_i}(t) = \sum_{j \in hp(i)} (1 + \lfloor t/T_j \rfloor) C_j \quad .$$

Le principe est le même que pour le calcul de L_i, mais en fixant le nombre exact d'activations de τ_i. L'idée est de comparer l'addition de ces trois termes, mais ici avec la longueur t de l'intervalle fermé $[0, t]$ car on veut prendre en compte les activations de tâches plus prioritaires qui arriveraient en t. Si le résultat est plus grand que t, alors $w_{i,q}$ est supérieur et l'argument est reconduit récursivement, jusqu'à ce que l'on trouve une valeur égale à la précédente. Formellement, $w_{i,q}$ est le point fixe de l'équation récursive convergente suivante :

$$\begin{cases} w_{i,q}^{(1)} = B_i + qC_i \\ w_{i,q}^{(m+1)} = B_i + \overline{W_i}(w_{i,q}^{(m)}) + qC_i \end{cases} \tag{36}$$

Le calcul est arrêté lorsque $w_{i,q}^{(m)} = w_{i,q}^{(m+1)} = w_{i,q}$. De cette façon, nous trouvons la plus petite solution de l'équation (avec $max_{k \in lp(i)}\{C_k - 1\} = 0$ si $lp(i) = \varnothing$) :

$$w_{i,q} = max_{k \in lp(i)}\{C_k - 1\} + \sum_{j \in hp(i)} (1 + \lfloor w_{i,q}/T_j \rfloor) C_j + qC_i. \tag{37}$$

VIII.2.2. Faisabilité basée sur les pires temps de réponse

Le Théorème 21 p. 70, énonce déjà ce résultat. L'argumentation porte juste ici sur les calculs précédents de *level-i busy periods* pour établir en contexte discret la faisabilité d'un trafic non-concret τ par un algorithme non-préemptif non-oisif à priorités fixes.

Théorème 25 - *[Hansson et al. 1994], [Burns et al. 1995], [George et al. 1996]. Soit un référentiel d'ordonnancement $\Sigma=(\Upsilon,\Pi)$ ou Υ contient tout TPN et Π contient NP-PF, un algorithme d'ordonnancement non-préemptif, non-oisif à priorités fixes. $\forall(\tau \in \Upsilon)$, τ est NP-PF-faisable si et seulement si:*

$$\forall i \in [1, n], r_i \le D_i, \text{ où } r_i = max_{0 \le q \le Q_i}\{w_{i,q} + C_i - qT_i\} \text{ avec:}$$

$$w_{i,q} = max_{k \in lp(i)}\{C_k - 1\} + \sum_{j \in hp(i)} (1 + \lfloor w_{i,q}/T_j \rfloor) C_j + qC_i,$$

$$Q_i = \lceil L_i/T_i \rceil - 1 \ (Q_i \text{ étant aussi la valeur minimale telle que } w_{i,Q_i} + C_i \le (Q_i + 1)T_i).$$

Preuve. $w_{i,q}$ donne la taille maximale d'une *level-i busy period* contenant q activations de τ_i et correspondant à l'un des scénarii d'activation ω de τ du Lemme 21 p. 79. Cette condition est nécessaire pour *NP-PF* puisque ces scénarii sont des instanciations concrètes possibles de tout $\tau \in \Upsilon$ et que $\forall i \in [1, n]$ et $\forall q$, $r_{i,q} = w_{i,q} + C_i - qT_i \le D_i$ si par hypothèse τ est faisable par *NP-PF*. Elle est suffisante pour *NP-PF* car tout temps de réponse dans un scénario d'activation ne peut qu'augmenter en le ramenant au scénario correspondant du Lemme 21 p. 79. ❐

Ce test nécéssaire et suffisant pour *NP-PF* est pseudo-polynomial si $U \le c < 1$ (cf. chapitre IX.2.2. p. 89). Il n'est pas suffisant pour *PF* (la version préemptive de *NP-PF*) car les pires temps de réponse obtenus ici ne sont pas nécessairement supérieurs à ceux obtenus par *PF*. Les calculs de $w_{i,q}$ sont en effet différents (cf. chapitre VIII.2.1.2. p. 80). Nous ne pouvons rien en déduire concernant l'optimalité et allons voir au chapitre suivant que contrairement au cas préemptif (cf. Théorème 8 p. 38), il n'est pas possible de le simplifier en considérant uniquement *q=0* lorsque $\forall i \in [1, n]$, $T_i \ge D_i$.

VIII.2.3. Une optimalité limitée

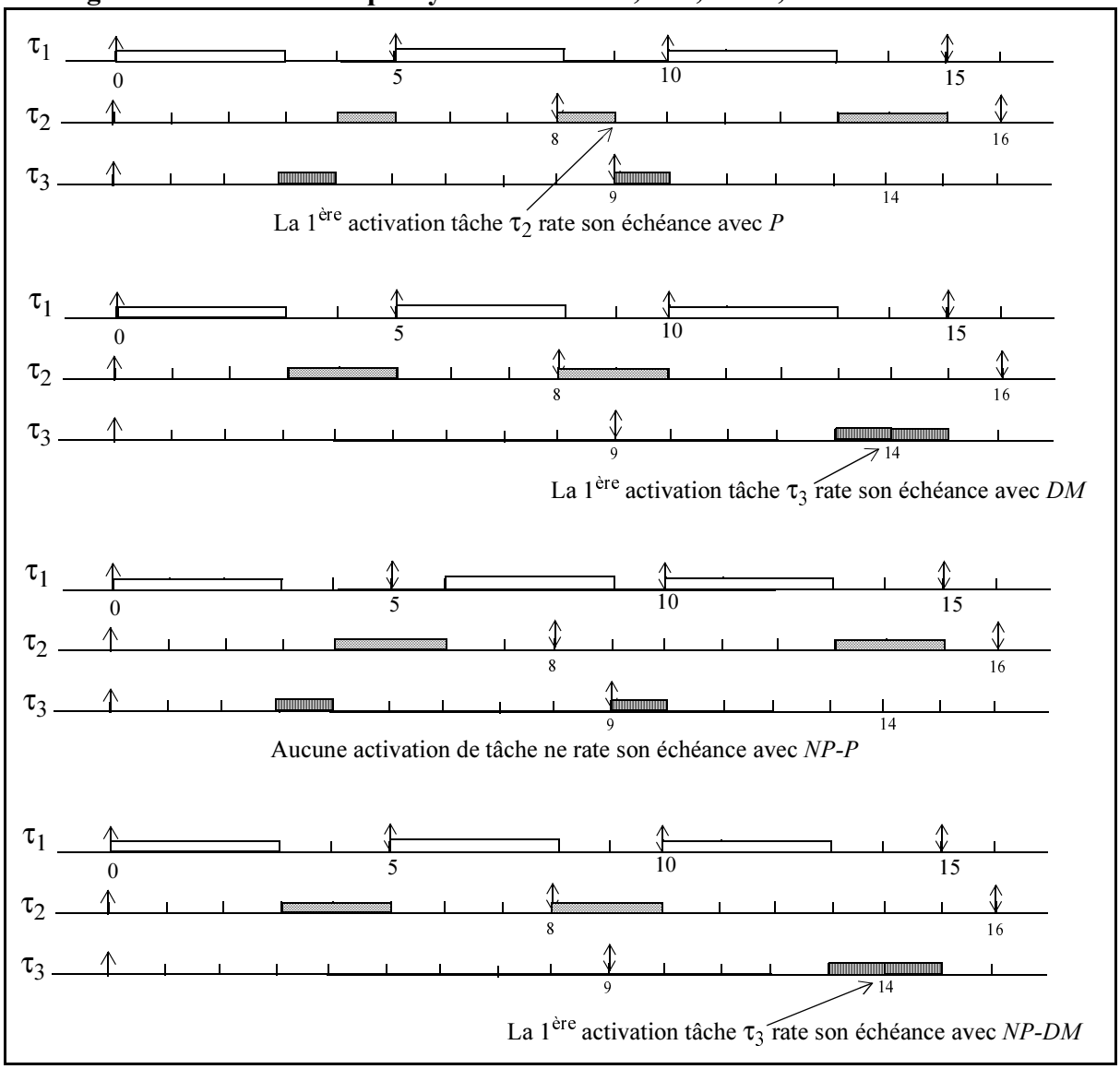
Contrairement à *NP-EDF/EDF* (cf. Théorème 23 p. 77), un algorithme à priorités fixes *NP-PF* n'est pas dominé, au sens de la Définition 35 p. 48, en termes de faisabilité et donc d'optimalité, par *PF* (sa version préemptive). En effet, nous venons de voir que l'absence de préemption n'agrave pas forcément les pires temps de réponse des tâches (seul moyen connu d'établir la faisabilité avec les priorités fixes) car les calculs de $w_{i,q}$ sont différents en préemptif et en non préemptif. Illustrons ceci par un simple trafic τ à 3 tâches :

$\tau_1(C_1=3, T_1=5, D_1=5)$, $\tau_2(C_2=2, T_2=8, D_2=8)$, $\tau_3(C_3=1, T_3=9, D_3=9)$.

Pour ce trafic, $D_i = T_i$, $\forall i \in [1, n]$ et L, la taille de la période occupée synchrone du processeur, vaut 15 quel que soit l'algorithme non-oisif utilisé (cf. équation (2) p. 17). Considérons l'ordonnancement de cette période occupée synchrone obtenu par quatre algorithmes non-oisifs (cf. Figure 17) :

- *P* (qui assigne les priorités dans l'ordre τ_1, τ_3, τ_2),
- *DM* (dans l'ordre τ_1, τ_2, τ_3),
- *NP-P* et *NP-DM* (les versions non-préemptives de *P* et *DM*).

Figure 17 : Période occupée synchrone avec *P, DM, NP-P, NP-DM*



Cette figure montre que τ n'est pas faisable avec *DM*, qui assigne les priorités dans l'ordre τ_1, τ_2, τ_3 (cf. Théorème 7 p. 37). Comme *DM* est l'algorithme préemptif à priorités fixes optimal lorsque $D_i \leq T_i$, $\forall i \in [1, n]$, τ n'est donc pas faisable par aucun algorithme préemptif à priorités fixes, en particulier par *P* qui assigne les priorités dans l'ordre τ_1, τ_3, τ_2.

Dans le même temps, τ n'est pas faisable non plus par *NP-DM* (la version non-préemptive, non-oisive de *DM*) mais est faisable par l'algorithme *NP-P* (la version non-préemptive, non-oisive de *P*), puisqu'alors $r_1 = 4 \leq D_1$, $r_3 = 5 \leq D_3$ et $r_2 = 7 \leq D_2$ (cf. Théorème 25 p. 81).

Il est remarquable qu'un trafic faisable par un algorithme non-préemptif non-oisif *NP-P* ne soit faisable ni par *P*, sa version préemptive, ni par aucun algorithme préemptif car *DM* (l'algorithme préemptif optimal dans ce cas) ne peut l'ordonnancer. L'absence de préemption, globalement pénalisante, peut donc parfois avantager certains algorithmes à priorités fixes en termes de faisabilité et d'optimalité. Ceci est impossible avec *EDF* (cf. Théorème 23 p. 77).

Remarquons de plus que le scénario synchrone étant toujours le pire pour la tâche de plus faible priorité en non-préemptif non-oisif (cf. Lemme 21 p. 79), nous constatons sur la figure (ce qui est confirmé par le Théorème 25 p. 81) que le pire temps de réponse de τ_2 est atteint sur sa deuxième $r_{3,(q=1)} = 7$, et non sa première $r_{3,(q=0)} = 6$, activation. Il n'est donc pas possible en non-préemptif non-oisif de simplifier le calcul en ne retenant que la valeur $q=0$, comme en préemptif, lorsque $D_i \leq T_i$, $\forall i \in [1, n]$.

Si nous nous limitons maintenant aux algorithmes non-préemptifs non-oisifs à priorités fixes, voyons comment dans [George et al. 1996] notre co-auteur L. George adapte la procédure d'*Audsley*[1], optimale en contexte préemptif pour les trafics non-concrets (cf. chapitre IV.3.4.3. p. 40 pour l'algorithme de cette procédure). La seule différence avec le contexte préemptif vient de l'utilisation du Théorème 25 p. 81 au lieu du Théorème 22 p. 71.

Lemme 22 - *[George et al. 1996]. Soit τ un trafic non-concret et NP-PF un algorithme d'ordonnancement non-préemptif, non-oisif à priorités fixes. Avec NP-PF, décroître la priorité d'une tâche diminue ou laisse inchangé les pires temps de réponse des autres tâches de τ.*

Preuve. Soit $\tau_i \in \tau$ la tâche qui voit sa priorité décroître.

- Avant le changement de priorité : $lp(i)$ est l'ensemble des tâches de plus basse priorité que τ_i, et $hp(i)$ l'ensemble des tâches de plus haute priorité que τ_i.
- Après le changement de priorité : $lp'(i)$ est l'ensemble des tâches de plus basse priorité que τ_i, et $hp'(i)$ l'ensemble initial des tâches de plus haute priorité que τ_i.

Après le changement de priorités, le pire temps de réponse est identique pour les tâches de $hp(i) \cup lp'(i)$ et ne peut pas augmenter pour les tâches de $lp(i) \cap hp'(i)$ (cf. Théorème 25 p. 81). Pour ces dernières en effet, τ_i n'apparaît plus que comme une inversion de priorité possible, contre au moins une exécution avant le changement de priorité. Toutes les tâches de τ, sauf τ_i, étant considérées, le résultat est prouvé. ❐

Théorème 26 - *[George et al. 1996], [George 1998]. La procédure d'assignation de priorités, dite d'Audsley, est Σ-optimal dans un référentiel d'ordonnancement $\Sigma=(\Upsilon,\Pi)$ où Υ contient tout TPN et Π contient tout algorithme d'ordonnancement à priorités fixes non-préemptif, non-oisif.*

Preuve. Soit τ un trafic non-concret de Υ. *Audsley* procède par priorités décroissantes (cf. chapitre IV.3.4.3. p. 40). S'il ne trouve aucune tâche faisable à un niveau de priorité par le Théorème 25 p. 81, alors le trafic est trivialement non faisable.

1. nous simplifions ici la preuve proposée initialement.

Si par contre il trouve une tâche faisable à ce niveau alors il est possible de lui affecter cette priorité ou la remplacer (cf. Lemme 22) par toute tâche restante faisable à un niveau de priorité supérieur mais faisable aussi à ce niveau de priorité. Il est donc impossible de trouver une assignation de priorités faisable que ne pourrait pas trouver cette procédure. ❒

Comme nous l'avons déjà remarqué lors du chapitre V.2.1. p. 47, cette procédure n'est pas totalement satisfaisante car elle reste coûteuse et dominée ici par *NP-EDF* (cf. Théorème 23 p. 77). Elle correspond en fait à une amélioration d'une procédure optimale en $n!$ (avec n le nombre de tâches), qui testerait toutes les permutations possibles pour assigner les priorités.

Il est important de souligner que contrairement aux cas préemptif, et excepté ce résultat relativement coûteux, nous ne connaissons aucune procédure d'assignation de priorités fixes optimale, même dans un référentiel d'ordonnancement restreint, en contexte non-préemptif non-oisif pour les trafics non-concrets.

IX. Eléments de comparaison des performances des algorithmes

Ce chapitre examine brièvement l'impact de l'absence de préemption sur les résultats d'efficacité des algorithmes puis de coût des tests. Quelques applications numériques seront proposées au chapitre X.2. p. 100.

IX.1. Efficacité des algorithmes

IX.1.1. Efficacité / borne de faisabilité

Nous avons vu lors du chapitre VI.1.2. p. 52, des référentiels d'ordonnancement $\Sigma = (\Upsilon, \Pi)$ compatibles pour la faisabilité avec la relation d'équivalence en périodes et en échéances (cf. Définition 36 p. 52). Ceci permet de passer l'ensemble Υ au quotient par cette relation et de se munir d'une base caractérisée par les couples *(T,D)* de Υ pour exprimer tout trafic non-concret $\tau \in \Upsilon$ comme combinaison linéaire des vecteurs de la base. Il est alors possible d'utiliser des normes pour mesurer l'efficacité exacte des algorithmes d'ordonnancement de Π. En particulier N_{EDF}, une norme valide appelée critère d'optimalité (cf. Théorème 17 p. 54), permet de mesurer $\alpha_{N_{EDF}}(P) = \varepsilon(P)$, l'efficacité exacte de tout algorithme d'ordonnancement $P \in \Pi$ en présence de *EDF* (avec $\varepsilon(EDF) = 1$).

En présence d'algorithmes non-préemptifs, la faisabilité de deux trafics non-concrets $(\tau, \tau') \in \Upsilon^2$ équivalents en périodes et en échéances n'est plus similaire en raison des inversions de priorités différentes possibles (cf. chapitre VIII. p. 72). Passer Υ au quotient par cette relation conduit alors à des conditions suffisantes relativement pessimistes sur la faisabilité des trafics puisque les inversions de priorités sont alors maximisées (en agrégeant les tâches caractérisées par les mêmes couples *(T,D)*, on allonge les inversions de priorités).

Les minorations proposées en préemptif sur l'efficacité des algorithmes à priorités fixes restent cependant valables (cf. [Leboucher 1998]). Celles-ci laissant parfois une large marge de manoeuvre et *NP-EDF* étant Σ-optimal en présence de trafics non-concrets et d'algorithmes non-préemptifs non-oisifs (cf. Théorème 23 p. 77), vérifions que l'avantage constaté dans certain cas préemptifs à utiliser *EDF* face aux priorités fixes (cf. chapitre VI.1.3. p. 55) reste significatif ici. Pour cela, sans passer Υ au quotient par la relation d'équivalence en périodes et en échéances, nous allons juste utiliser N_{EDF} pour en déduire des bornes de faisabilité à interpréter. En effet, pour tout $\tau \in \Upsilon$ et tout algorithme (non-préemptifs y compris), nous avons la condition nécessaire :

$$1 \geq sup_{t \in \mathbb{R}^+} \left\{ \frac{h(t)}{t} \right\} = N_{EDF}(\tau) \text{ (cf. Lemme 1 p. 16).} \quad (38)$$

$N_{EDF}(\tau)$ ne peut donc être supérieur à 1 si τ est faisable par *EDF*. Il est donc possible de calculer une borne de faisabilité dans un référentiel d'ordonnancement Σ pour un algorithme non-préemptif $P \in \Pi$ comme suit (cf. Définition 38 p. 53):

$$\alpha_{N_{EDF}}(P) = sup\{\alpha / (\forall \tau \in \Upsilon, N_{EDF}(\tau) \leq \alpha \Rightarrow P(\tau))\}. \quad (39)$$

Notons que nous ne pouvons pas en déduire[1] $\varepsilon(P)$ et que toute condition suffisante basée sur N_{EDF} donne alors uniquement une minoration de cette borne de faisabilité pour *P*. Quelques tendances sur l'évolution de telles minorations sont examinées ici.

1. *EDF* est Σ-optimal en présence d'algorithmes non-préemptifs et l'éq. (38) est un test nécéssaire et suffisant pour *EDF* (cf. Théorème 12 p. 46). Une conjecture est donc que la borne de faisabilité basée sur N_{EDF} est la meilleure possible pour les algorithmes non-préemptifs en présence de *EDF*. Notons cependant que les preuves des propriétés énoncées par le Théorème 16 p. 53, ne sont plus valables sans la relation d'équivalence en périodes et en échéances. Nous ne pouvons donc pas affirmer que N_{EDF} est le critère d'optimalité, qui donnerait la meilleure borne dans ce cas.

IX.1.2. *EDF/NP-EDF*

Compte-tenu des inversions de priorités possibles, $\delta D = max(D_i)/min(D_i)$ la dispersion en échéances devrait pénaliser $\alpha_{N_{EDF}}(NP\text{-}EDF)$ en comparaison de $\alpha_{N_{EDF}}(EDF)$. Pour préciser cette intuition, considérons le test nécéssaire et suffisant pour tout trafic non-concret $\tau \in \Upsilon$ par *NP-EDF* (cf. Théorème 23 p. 77) :

$$\forall t \geq 0,\quad t \geq max_{D_i > t}\{C_i - 1\} + h(t) \text{ (avec } max_{D_i > t}\{C_i - 1\} = 0 \text{ si } \nexists i\text{: } D_i > t),$$

$$\text{c.à.d. } \forall t \geq 0, \qquad \frac{h(t)}{t} \leq 1 - \frac{max_{D_i > t}\{C_i - 1\}}{t}. \tag{40}$$

En se limitant à $min(D_i)$, le premier point de discontinuité pertinent, nous obtenons le test suffisant suivant pour tout trafic non-concret $\tau \in \Upsilon$ par *NP-EDF* :

$$N_{EDF}(\tau) \leq max\left(0, \frac{min(D_i) - max_{D_i > min(D_i)}\{C_i - 1\}}{min(D_i)}\right). \tag{41}$$

Le cas $\forall i$, D_i=D.

Dans ce cas $\alpha_{N_{EDF}}(NP\text{-}EDF) = \varepsilon(EDF) = 1$ car trivialement il ne peut y avoir d'inversions de priorités. Ceci est confirmé par l'éq. (40) (un test nécéssaire et suffisant pour *NP-EDF*) puisque $\forall t \geq min(D_i)$, $max_{D_i > t}\{C_i - 1\} = 0$. En fait dans ce cas particulier, à fortiori vrai dans le cas homogène (c.à.d. $\forall i$, D_i=D, T_i=T), le scénario d'activation pire cas redevient comme pour *EDF* le scénario synchrone avec *NP-EDF*.

Le cas de Liu & Layland (c.à.d. $\forall i$, $D_i = T_i$).

Dans ce cas, plus $\delta T = max(T_i)/min(T_i)$ (la dispersion en période) augmente, plus $\alpha_{N_{EDF}}(NP\text{-}EDF)$ diminue. Il est en effet facile d'identifier un trafic $\tau \in \Upsilon$ faisable par *EDF* mais non faisable par *NP-EDF*, par exemple si un scénario d'activation ω de τ conduit à un facteur de blocage suffisant sur la tâche de plus petite période. Avec les mêmes durées d'exécution, si nous modifions Υ en augmentant les périodes, sauf le $min(T_i)$, alors δT augmente et :

- le scénario ω est toujours trivialement faisable par *EDF* et non faisable par *NP-EDF*.
- N_{EDF} diminue car N_{EDF}=N_U=$\sum_n C_i/T_i$ dans ce cas (cf. Théorème 2 p. 30). Il vient de l'éq. (39) que $\alpha_{N_{EDF}}(NP\text{-}EDF) = \alpha_{N_U}(NP\text{-}EDF)$ diminue aussi.

Par le raisonnement inverse, il est remarquable que $min(T_i)$ a une influence forte sur $\alpha_{N_{EDF}}(NP\text{-}EDF)$ (même si toutes les autres tâches ont des grandes périodes). En effet plus ce terme est petit devant les autres périodes, plus le facteur de blocage (c.à.d. la durée d'exécution) nécessaire pour rendre le trafic non faisable par *NP-EDF* est petit. Une seule tâche ayant une petite période suffit à faire fortement diminuer $\alpha_{N_{EDF}}(NP\text{-}EDF)$.

Le cas $\forall i$, $D_i \geq T_i$.

Dans ce cas, plus les échéances relatives des tâches (en particulier le $min(D_i)$) augmentent en comparaison des périodes, plus $\alpha_{N_{EDF}}(NP\text{-}EDF)$ tend vers 1. En effet, pour tout trafic τ faisable par *EDF*, l'éq. (41) (un test suffisant pour *NP-EDF*) tend vers $N_{EDF}(\tau) \leq 1$ lorsque $min(D_i)$ augmente en comparaison des périodes (et donc de $max_{D_i > min(D_i)}\{C_i - 1\}$) puisque $\forall i, C_i \leq T_i$ est une condition nécessaire triviale pour que τ reste faisable par EDF. Il vient, de l'éq. (39), que $\alpha_{N_{EDF}}(NP\text{-}EDF)$ tend aussi vers 1.

Le cas $\forall i, D_i \leq T_i$.

Dans ce cas, certains trafics $\tau \in \Upsilon$ peuvent vérifier $max_{D_i > min(D_i)}\{C_i - 1\} > min(D_i)$. Le test suffisant donnée par l'éq. (41) n'a donc aucun sens. En reprenant le raisonnement tenu dans le cas Liu & Layland, mais en utilisant la formule de N_{EDF} (cf. Théorème 17 p. 54) au lieu de N_U, il vient que $\alpha_{N_{EDF}}(NP\text{-}EDF)$ diminue lorsque la dispersion en échéances ou en périodes augmente. Ceci est d'autant plus marqué que le $min(D_i)$ diminue.

Le cas général où les échéances relatives ne sont pas liées aux périodes.

Excepté les tendances décrites ci-dessus, nous n'avons pas d'interprétation claire dans le cas général. En particulier, compte-tenu du terme négatif de l'éq. (41), nous n'avons pas de minoration sur $\alpha_{N_{EDF}}(NP\text{-}EDF)$. Notons cependant qu'il reste toujours des trafics $\tau \in \Upsilon$ faisables par *EDF* et *NP-EDF*, $\alpha_{N_{EDF}}(NP\text{-}EDF)$ (le terme gauche de l'éq. (39)) ne peut donc jamais s'annuler.

En conclusion, la borne de faisabilité $\alpha_{N_{EDF}}(NP\text{-}EDF)$ est donc pénalisée par les inversions de priorités possibles lorsque la dispersion augmente et/ou que certaines échéances sont plus petites que les périodes. Voyons maintenant ce qu'il en est des priorités fixes.

IX.1.3. Les priorités fixes

Les résultats du chapitre VIII. p. 72, nous donnent quelques indications :

- *NP-EDF* est Σ-optimal en présence de trafics non-concrets et d'algorithmes non-préemptifs non-oisifs (cf. Théorème 23 p. 77). Il vient que $\alpha_{N_{EDF}}(NP\text{-}EDF) \geq \alpha_{N_{EDF}}(NP\text{-}PF)$ dans un tel référentiel d'ordonnancement Σ (avec *NP-PF*, un algorithme non-préemptif non-oisif à priorités fixes quelconque). Nous allons vérifier si cette propriété de *NP-EDF* conduit à un avantage aussi significatif qu'en préemptif en terme d'efficacité.
- pour la même assignation de priorités fixes, il est possible d'être faisable en non-préemptif sans l'être en préemptif (cf. chapitre VIII.2.3. p. 82). Nous ne pouvons donc pas énoncer que $\alpha_{N_{EDF}}(PF) \geq \alpha_{N_{EDF}}(NP\text{-}PF)$ (avec *PF* la version préemptive de *NP-PF*)

Malgré ce dernier résultat, la dispersion devrait encore fortement jouer en faveur des algorithmes préemptifs. Précisons ceci en nous inspirant de la démarche suivie pour *NP-EDF,* mais en utilisant $N_U \leq N_{EDF}$ (cf. Théorème 12 p. 46). Le chapitre VI.1.3. p. 55, nous a permis d'en déduire un minorant sur l'efficacité des algorithmes préemptifs à priorités fixes. Le même raisonnement, partant ici du Théorème 25 p. 81, nous conduit au test suffisant suivant en non-préemptif (avec *lp* la priorité fixe la plus faible et $max_{k \neq lp}\{C_k - 1\} = 0$ s'il n'y a qu'une priorité) :

$$N_U \leq max\left(0, \frac{min(D_i) - max_{k \neq lp}\{C_k - 1\}}{max(T_i) + min(D_i)}\right). \quad (42)$$

Compte-tenu du terme négatif, ce résultat (à comparer avec l'équation (22) p. 56 en préemptif) ne peut conduire à une minoration pertinente de $\alpha_{N_U}(NP\text{-}PF)$ dans le cas général. Voyons cependant, de façon très similaire à ce qui à été dit pour *NP-EDF*, comment il est possible d'interpréter certains cas particuliers.

Le cas homogène (c.à.d. $\forall i$, $D_i=D$, $T_i=T$).

Trivialement dans ce cas il est encore impossible d'être non faisable par *NP-PF* et faisable par *PF*. On a donc $\alpha_{N_U}(NP\text{-}PF) = \varepsilon(PF) = \varepsilon(EDF) = 1$ (cf. chapitre VI.1.3.2.b p. 57).

Le cas de Liu & Layland (c.à.d. $\forall i$, $D_i = T_i$).

Dans ce cas, plus $\delta T = max(T_i)/min(T_i)$ (la dispersion en période) augmente, plus $\alpha_{N_U}(NP\text{-}PF)$ diminue face à $\varepsilon(PF)$. Il est en effet facile d'identifier un trafic $\tau \in \Upsilon$ faisable par *PF* mais non faisable par *NP-PF*. Par exemple avec *PF=RM*, l'algorithme à priorités fixes préemptif optimal dans de cas, si un scénario d'activation ω de τ conduit à un facteur de blocage suffisant sur la tâche la plus prioritaire, c.à.d. de plus petite période. Avec les mêmes durées d'exécution, si nous modifions Υ en augmentant les périodes, sauf le $min(T_i)$, alors δT augmente et :

- le trafic modifié est toujours trivialement non faisable par *NP-RM* et faisable par *RM.*
- $N_U = \sum_n C_i/T_i$ diminue. $\alpha_{N_U}(NP\text{-}RM)$ ne peut donc que diminuer par l' éq. (39).

Rappelons que les bornes $\alpha_{N_U}(NP\text{-}RM)$ et $\alpha_{N_{EDF}}(NP\text{-}RM)$ sont égales dans ce cas (cf. Théorème 2 p. 30). Ce résultat est donc valable avec la meilleure borne de faisabilité en notre connaisance (cf. chapitre IX.1.1. p. 85). Comme pour *NP-EDF*, la diminution de cette borne est fortement influencée par la dispersion alors que l'efficacité de *RM* ne peut descendre en dessous de $n(2^{1/n}-1)$ d'après [Liu and Layland 1973]. Notons que le raisonnement est identique si nous diminuons $min(T_i)$. Une seule tâche avec une petite période influe donc fortement sur l'efficacité de *NP-RM.*

Le cas $\forall i$, $D_i \geq T_i$.

Dans ce cas, plus les échéances relatives des tâches (en particulier $min(D_i)$) augmentent en comparaison des périodes, plus $\alpha_{N_U}(NP\text{-}RM)$ tend vers 1. En effet, pour tout trafic τ faisable par *PF*, l'éq. (42) (un test suffisant pour *NP-PF*) tend alors vers $N_U(\tau) \leq min(D_i)/(max(T_i) + min(D_i))$ lorsque $min(D_i)$ augmente en comparaison des périodes (et donc de $max_{k \neq lp}\{C_k - 1\}$) puisque $\forall i$, $C_i \leq T_i$ est une condition nécessaire pour que τ reste faisable par *PF*. L'éq. (42) tend donc vers $N_U(\tau) \leq 1$ lorsque $min(D_i)$ augmente aussi face à $max(T_i)$. Il vient, de l'éq. (39), que $\alpha_{N_U}(NP\text{-}PF)$ tend aussi vers 1

L'intuition est confirmée, le choix de l'algorithme (préemptif/non-préemptif, priorités fixes/dynamiques) perd de son intérêt lorsque les échéances relatives sont grandes face aux périodes.

Le cas $\forall i$, $D_i \leq T_i$.

Dans ce cas, certains trafics $\tau \in \Upsilon$ peuvent vérifier $max_{k \neq lp}\{C_k - 1\} > min(D_i)$. Le test suffisant donné par l'éq. (42) n'a donc aucun sens. En reprenant le raisonnement tenu dans le cas Liu & Layland, mais en utilisant *DM* au lieu de *RM* et la formule de N_{EDF} (cf. Théorème 17 p. 54) au lieu de N_U, il vient que $\alpha_{N_{EDF}}(NP\text{-}DM)$ diminue lorsque la dispersion en échéances ou en périodes augmente. Ceci est d'autant plus marqué que le $min(D_i)$ diminue.

Le cas général où les échéances relatives ne sont pas liées aux périodes.

Excepté les tendances décrites ci-dessus, nous n'avons pas d'interprétation claire dans le cas général. En particulier, compte-tenu du terme négatif de l'éq. (42), nous n'avons pas de minoration sur $\alpha_{N_{EDF}}(NP\text{-}PF)$. Notons cependant qu'il reste toujours des trafics faisables par *NP-PF*, $\alpha_{N_{EDF}}(NP\text{-}PF)$, le terme gauche de l'éq. (39), ne peut donc jamais s'annuler.

En conclusion, les bornes de faisabilité $\alpha_{N_{EDF}}(NP\text{-}PF)$ et $\alpha_{N_{EDF}}(NP\text{-}EDF)$ sont donc toutes les deux très pénalisées par les inversions de priorités possibles lorsque la dispersion augmente et/ou que certaines échéances sont plus petites que les périodes. Bien que *NP-EDF* soit Σ-optimal en présence trafics non-concrets et d'algorithmes non-préemptifs non-oisifs (cf. Théorème 23 p. 77), il semble donc que l'avantage constaté en préemptif à utiliser *EDF* par rapport aux priorités fixes (cf. chapitre VI.1.3. p. 55) ne soit pas significatif ici. Quelques applications numériques seront proposées au chapitre X.2. p. 98, pour illustrer notre propos.

IX.2. Coût des conditions de faisabilité

Contrairement à l'efficacité des algorithmes, le coût des tests n'est pas modifié par le passage du préemptif au non-préemptif, non-oisif. En effet la modification des tests porte principalement sur la prise en compte d'un terme additionel, le facteur de blocage. Nous reprenons donc simplement les résultats préemptifs du chapitre VI.2.2. p. 62, en précisant l'impact de ces facteurs de blocage sur le coût des tests.

Notons tout d'abord que le coût du calcul de L, la taille de la période occupée du processeur servant à borner les intervalles d'études, est identique au cas préemptif (cf. chapitre VI.2.2.2. p. 62). En effet L ne dépend pas de l'algorithme choisi (préemptif/non-préemptif, à priorités fixes/dynamiques) dès lors que celui-ci est non-oisif mais uniquement du scénario d'activation (cf. Lemme 2 p. 17).

IX.2.1. Majorant sur le coût de la faisabilité seule par *NP-EDF*

D'après le Théorème 23 p. 77, $\forall t \in S$ il faut tester le prédicat $h(t) + B(t) \leq t$ pour établir la faisabilité d'un trafic par *NP-EDF*. $C_{NP\text{-}EDF,\,(\forall t \in S,\, P(t))}$, le coût de ce calcul, est donc égal à $C_{EDF,\,(\forall t \in S,\, P(t))}$, le coût du même calcul dans le cas préemptif (cf. chapitre VI.2.2.3. p. 62), en tenant compte du terme additionel $B(t)$ sur chaque point de discontinuité. Comme $B(t) = max_{D_i > t}\{C_i - 1\}$ est en *O(n)*, l'ordre de grandeur de ces coûts est le même.

$$C_{NP\text{-}EDF,\,(\forall t \in S,\, P(t))} \leq O(n^2).$$

IX.2.2. Majorant sur le coût du calcul des r_i par un algorithme à priorités fixes

D'après le Théorème 25 p. 81, $\forall i \in [1, n]$ il faut tester le prédicat $r_i = max_{q \leq Q_i}\{w_{i,q} + C_i - qT_i\} \leq D_i$, $w_{i,q} = qC_i + \sum_{j \in hp(i)}(1 + \lfloor w_{i,q}/T_j \rfloor)C_j + B_i$ pour calculer le pire temps de réponse de la tâche τ_i d'un trafic général par un algorithme non-préemptif, non-oisif à priorités fixes.

$C_{NP\text{-}FP,\,(\forall i \in [1, n],\, r_i \leq D_i)}$, le coût de ce calcul, est donc égal à $C_{FP,\,(\forall i \in [1, n],\, r_i \leq D_i)}$, le coût du même calcul dans le cas préemptif (cf. chapitre VI.2.2.4. p. 63), mais en tenant compte du terme additionel B_i sur chaque itération. Comme $B_i = max_{k \in lp(i)}\{C_k - 1\}$ est en *O(n)*, l'ordre de grandeur de ces coûts est le même.

$$C_{NP\text{-}FP,\,(\forall i \in [1, n],\, r_i \leq D_i)} \leq O(n^3).$$

Rappelons cependant que contrairement aux cas préemptif avec *DM* nous ne connaissons aucune procédure d'assignation de priorités fixes optimale et à faible coût en contexte non-préemptif non-oisif pour les trafics non-concrets, même dans un référentiel d'ordonnancement restreint (cf. chapitre VIII.2.3. p. 82). Si nous utilisions la procédure d'*Audsley*, optimale dans le cas général, il faudrait alors mutiplier par $O(n^2)$ la complexité des résultats (cf. Théorème 26 p. 83).

IX.2.3. Majorant sur le coût du calcul des r_i par *NP-EDF*

D'après le Théorème 24 p. 78, $\forall i \in [1, n]$ il faut tester le prédicat $r_i = max_{a \in S}\{C_i,\ L_i(a) + C_i - a\} \leq D_i$ pour calculer le pire temps de réponse de la tâche τ_i d'un trafic général par *NP-EDF*. Avec :

$$L_i(a) = B(a + D_i) + \sum_{\substack{j \neq i \\ D_j \leq a + D_i}} min\left\{1 + \left\lfloor \frac{L_i(a)}{T_j} \right\rfloor, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor\right\} C_j + \left\lfloor \frac{a}{T_i} \right\rfloor C_i .$$

$C_{NP\text{-}EDF,\ (\forall i \in [1, n],\ r_i \leq D_i)}$, le coût de ce calcul, est donc égal à $C_{EDF,\ (\forall i \in [1, n],\ r_i \leq D_i)}$, le coût du même calcul dans le cas préemptif (cf. chapitre VI.2.2.5. p. 63), en tenant compte du terme additionel $B(a + D_i)$ sur chaque itération. Comme $B(a + D_i) = max_{D_j > a + D_i}\{C_j - 1\}$ est en $O(n)$, l'ordre de grandeur de ces coûts est le même.

$$C_{NP\text{-}EDF,\ (\forall i \in [1, n],\ r_i \leq D_i)} \leq O(n^3) .$$

Partie D : Applications numériques et Synthèse

Cette partie illustre les résultats d'efficacité des algorithmes par quelques applications numériques. Elle propose ensuite, lors du chapitre XI. p. 103, une synthèse des résultats en revenant sur les propriétés spécifiées dans notre cahier des charges.

X. Applications numériques

X.1. Le cas préemptif

Le chapitre VI.1. p. 51, examine l'efficacité des algorithmes d'ordonnancement préemptifs. Soit $\Sigma = (\Upsilon,\Pi)$, un référentiel d'ordonnancement où Υ est caractérisé par des couples[1] (T,D) et Π par des algorithmes compatibles pour la faisabilité avec la propriété d'équivalence en période et en échéance. Dans un tel référentiel Σ, rappelons que *EDF* est Σ-optimal et que N_{EDF} est un critère d'optimalité. Il suit que :

- l'efficacité de *EDF* est toujours maximale car $\alpha_{N_{EDF}}(EDF) = \varepsilon(EDF) = 1$,
- pour tout autre algorithme $P \in \Pi$, $\alpha_{N_{EDF}}(P) = \varepsilon(P) \leq \varepsilon(EDF)$,
- pour toute norme valide N, $\alpha_N(P) \leq \alpha_{N_{EDF}}(P) = \varepsilon(P) \leq 1$.

1. Rappelons que les couples *(T,D)* d'un référentiel d'ordonnancement Σ ne limitent pas le nombre de tâches mais imposent à celles-ci les valeurs des périodes et des échéances relatives (cf. Définition 27 p. 22).

Nous allons maintenant comparer le calcul exact[1] de l'efficacité d'algorithmes d'ordonnancement à priorités fixes avec les minorations obtenues par les théorèmes d'efficacité. Rappelons que ces théorèmes ont pour objectif de prévoir les variations de l'efficacité lorsque le référentiel d'ordonnancement varie. Le premier théorème d'efficacité (établi dans [Hermant et al. 1996] par notre co-auteur L. Leboucher, cf. Théorème 18 p. 56) s'applique à tout algorithme d'ordonnancement à priorités fixes $P \in \Pi$. Nous l'appellerons donc "minoration de $\varepsilon(P)$" dans la suite. Le deuxième théorème se spécialise pour l'algorithme *DM* (cf. Théorème 19 p. 59). Nous l'appellerons donc "minoration de $\varepsilon(DM)$".

La Table 12 de l'annexe A, p. 117, détaille les calculs de normes, d'efficacité et de minorations de l'efficacité sur quelques référentiels d'ordonnancement particuliers. Au delà de ces calculs, nous allons utiliser ici deux techniques pour tenter d'illustrer la "dynamique" des variations de l'efficacité lorsque le référentiel d'ordonnancement varie :

- modification du référentiel initial par une augmentation de $\delta T = Max(T)/Min(T)$, la dispersion en période de Υ, (cf. chapitre X.1.1. p. 94).
- modification du référentiel initial, à périodes constantes, par décalage des échéances relatives (cf. chapitre X.1.2. p. 96).

X.1.1. Influence de la dispersion

Commençons par un exemple simple (cf. Figure 18) avec deux couples $(T_1,D_1=T_1)$ et $(T_2,D_2=T_2)$. Nous sommes dans le cas [Liu and Layland 1973] où $N_{EDF}(\tau)=N_U(\tau)=N'_U(\tau)$ (cf. Théorème 2 p. 30, pour N_U et N_{EDF} et Equation (20) p. 56, pour N_U et N'_U). Intéressons nous aux mesures d'efficacité (en ordonnée) lorsque δT augmente (en abscisse) :

- $\varepsilon(EDF) = 1$ dans tous les cas comme attendu (cf. Théorème 17 p. 54).
- La minoration de $\varepsilon(P)$ décroît comme prévu avec la dispersion (cf. chapitre VI.1.3.2.b p. 57). Rappelons que cette minoration considère tous les algorithmes d'ordonnancement à priorités fixes $P \in \Pi$. Ceci est confirmé par $\varepsilon(X)$, le calcul exact de l'efficacité d'un "mauvais" algorithme qui assigne les priorités dans l'ordre inverse de *RM*, l'algorithme optimal dans ce cas (cf. chapitre IV.3.2. p. 35). On voit bien que les variations de la minoration de $\varepsilon(P)$ et de $\varepsilon(X)$ sont proches.
- Le calcul exact de $\varepsilon(DM)$ montre cependant que la minoration de $\varepsilon(P)$ n'est pas représentative si l'on utilise l'algorithme *DM=RM*, optimal parmi les algorithmes à priorités fixes dans ce cas. Notons que la minoration sur $\varepsilon(DM)$ obtenue par le deuxième théorème d'efficacité donne 1/2, comme prévu dans ce cas (cf. chapitre VI.1.4.2. p. 59). Celle-ci est plus proche de $n(2^{1/n}-1)$, la borne de [Liu and Layland 1973]. En particulier, ces deux valeurs ne varient pas avec la dispersion.

Ces tendances sont confirmées par la Figure 19, qui étend l'exemple à 5 couples $(T,D=T)$ dont les valeurs T sont réparties régulièrement entre T_{min} et T_{max}. Les seules différences sont :

- la borne de [Liu and Layland 1973] diminue car elle dépend du nombre de couples (T,D),
- le calcul exact de $\varepsilon(DM)$ n'est plus égal à 1 lorsque δT a une valeur entière. Ceci s'explique car les périodes ne peuvent plus être ici multiples simultanément de T_{max} (sauf bien sur lorsque δT=1). Cette remonté possible de l'efficacité, remarquée par [Liu and Layland 1973], illustre l'intérêt d'imposer de telles relations entre les périodes avec les algorithmes à priorités fixes.

1. basée sur la Définition 38 p. 53, c.à.d. $sup\{\alpha/(\forall\tau\in\Upsilon, N_{EDF}(\tau)\le\alpha\Rightarrow P(\tau))\}$, *la* ***$N_{EDF}$-efficacité*** *de P dans* Σ (cf. [Hermant et al. 1996] pour la procédure de calcul utilisée).

Figure 18 : Influence de la dispersion dans le cas de Liu et Layland avec 2 couples *(T,D)*

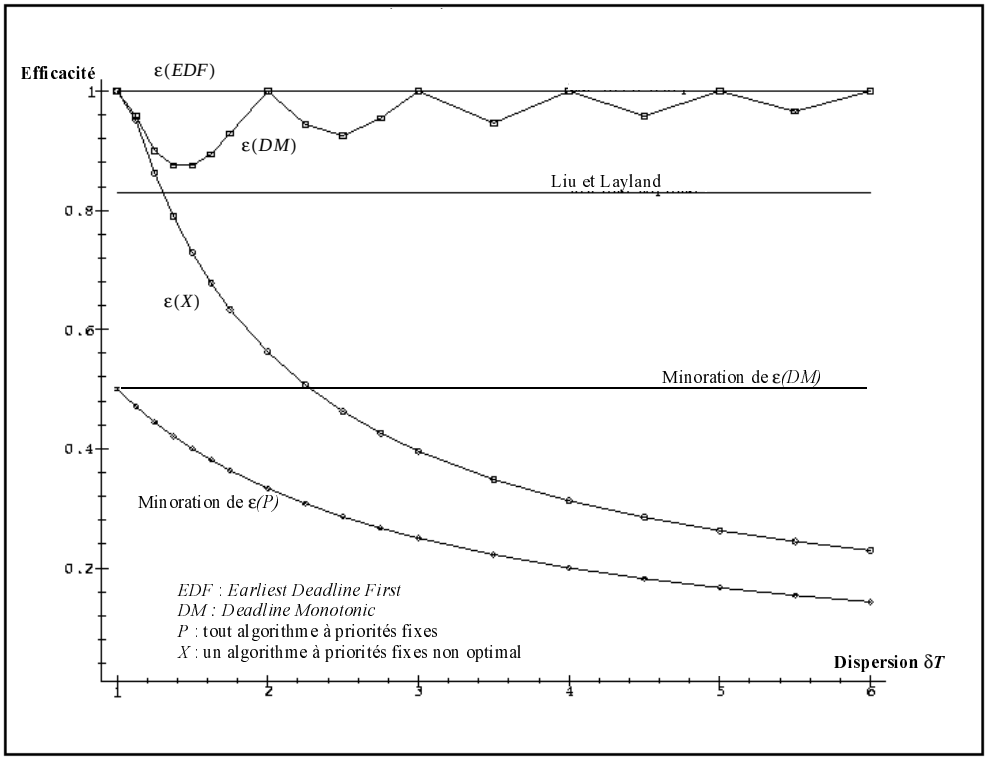
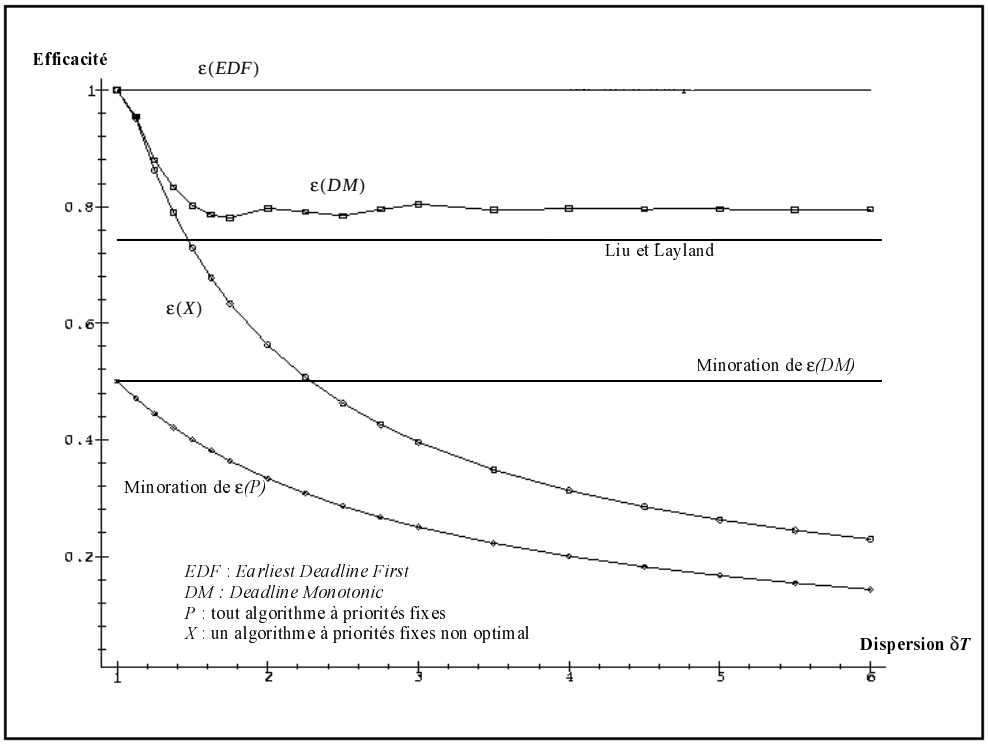


Figure 19 : Influence de la dispersion dans le cas de Liu et Layland avec 5 couples *(T,D)*



X.1.2. Influence des échéances relatives

Pour illustrer l'impact des échéances relatives sur l'efficacité des algorithmes, nous considérons maintenant des référentiels d'ordonnancement $\Sigma = (\Upsilon,\Pi)$ où les périodes des couples *(T,D)* représentées dans Υ sont constantes. Nous faisons varier par contre les échéances relatives simultanément dans le même sens (augmentation ou diminution). Plus précisement, l'ordonnée des courbes donne comme précédemment des mesures d'efficacité. L'abscisse des courbes donne ici, pour toute valeur de *x*, l'échéance moyenne moins la période moyenne parmi ces couples *(T,D)*. Il suit que vers la gauche des figures les échéances relatives diminuent en comparaison des périodes. Vers la droite elles augmentent. Encore une fois, au delà des valeurs obtenues l'objectif est avant tout de vérifier ici les comportements prévus par les théorèmes d'efficacité.

Commençons (cf. Figure 20) par mesurer les variations d'efficacité de l'algorithme *DM* sur un exemple à cinq couples *(T,D)*. Nous imposons $\delta T = \delta D = 2$, c.à.d. des dispersions constantes[1]. $x = 0$ signifie ici que $\forall i, D_i = T_i$ (le cas de Liu & Layland), $x < 0$ signifie $\forall i, D_i < T_i$ et $x > 0$ signifie $\forall i, D_i > T_i$. De plus, $x < -15$ signifie que toutes les échéances relatives sont inférieures à la plus petite période et $x > 30$ signifie que toutes les échéances relatives sont supérieures à la plus grande période. Nous remarquons que :

- pour tout $x \geq 0$, $\varepsilon(DM) = \alpha_{N_{edf}}(DM) = \alpha_{N'_U}(DM) = \alpha_{N_U}(DM)$ car ces normes valides sont équivalentes lorsque $\forall i, D_i \geq T_i$ (cf. Théorème 2 p. 30, pour N_U et N_{EDF} et Equation (20) p. 56, pour N_U et N'_U). Il apparaît surtout que $\varepsilon(DM)$ tend vers 1 lorsque les *x* augmentent, c.à.d. lorsque les échéances deviennent grandes en comparaison des périodes. Ceci est bien le comportement prévu et vérifié par la minoration sur $\varepsilon(P)$ du premier théorème d'efficacité (cf. chapitre VI.1.3.2.b p. 57) et celle sur $\varepsilon(DM)$ (cf. chapitre VI.1.4.2. p. 59). Notons que cette deuxième minoration est moins pessimiste car car elle se spécialise pour *DM*. Elle est en particulier toujours supérieure à 1/2 dès lors que $\forall i, D_i \geq T_i$.
- lorsque $x \leq 0$, le calcul exact de $\varepsilon(DM)$ passe par une plage minimum puis augmente rapidement dès que les échéances relatives deviennent toutes inférieures aux périodes. Ceci n'est pas décelable avec $\alpha_{N_U}(DM)$. Par contre avec $\alpha_{N'_U}(DM)$, dont nous déduisons la minoration sur $\varepsilon(P)$ (cf. chapitre VI.1.3.2.a p. 56), nous observons une stabilisation de la minoration sur $\varepsilon(DM)$ lorsque les échéances relatives deviennent toutes inférieures aux périodes. $\alpha_{N'_U}(DM)$ est donc bien une meilleure approximation de $\varepsilon(DM)$ que $\alpha_{N_U}(DM)$ et $\varepsilon(DM)$ ne peut donc s'annuler. Notons encore une fois que la minoration spécialisée sur $\varepsilon(DM)$ est moins pessimiste que celle sur $\varepsilon(P)$. Elle tend vers $1/\delta T$ (égal ici à 1/2) quand on va vers la gauche de la figure (cf. chapitre VI.1.4.2. p. 59). Nous aurions pu la faire augmenter un peu plus en choisissant une dispersion en période δT inférieure à 2.

La Figure 21 examine le même exemple avec *X*, un "mauvais" algorithme d'ordonnancement qui assigne des priorités fixes dans l'ordre inverse de *DM*. Nous constatons bien que les mesures d'efficacité sur *X* sont moins bonnes que celles de *DM*, en particulier lorsque les échéances relatives sont faibles face aux périodes (zone où *DM* est optimal). La minoration $\varepsilon(P)$ obtenue par le premier théorème d'efficacité reste cependant valable car elle couvre tout algorithme à priorités fixes préemptif.

1. δT reste constant car car les valeurs des périodes sont fixes. Nous sommes par contre obligé de resserrer les valeurs des échéances relatives vers la gauche de la figure et les écarter vers la droite pour conserver δD=2.

Figure 20 : Influence des échéances sur ε*(DM)* à dispersion constante (δ*T*=δ*D*=2)

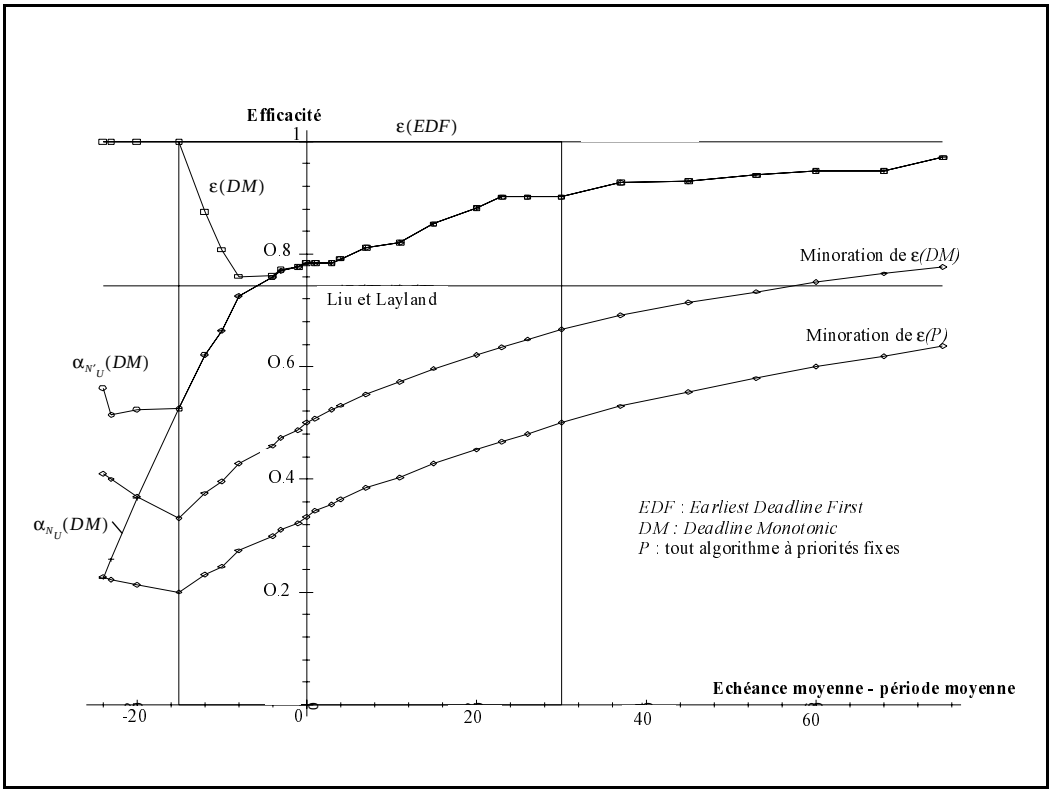
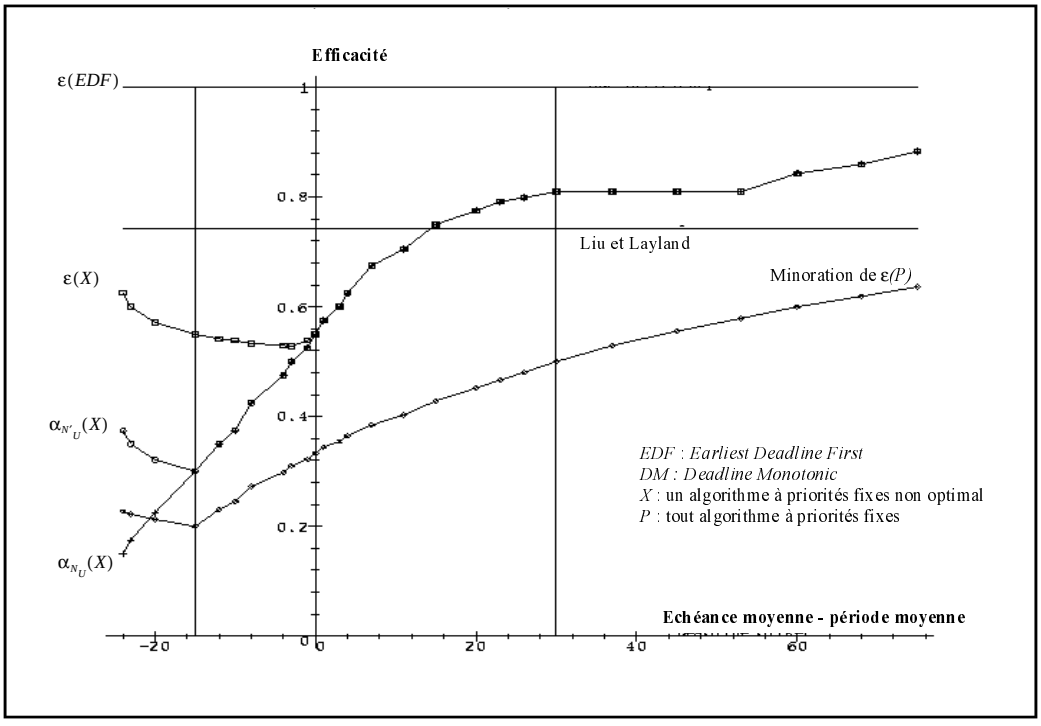


Figure 21 : Influence des échéances sur ε*(X)* à dispersion constante (δ*T*=δ*D*=2) avec *X*, un "mauvais" algorithme à priorités fixes



La Figure 22 considère un cas similaire au précédent. L'objectif est ici de ne pas passer par les valeurs $\forall i, D_i = T_i$ en x=0. Celles-ci favorisent en effet les algorithmes *RM* et *DM* dont l'efficacité ne peut alors descendre en dessous de $n(2^{1/n} - 1)$ en ce point central de la figure (cf. [Liu and Layland 1973]).

Pour simplifier l'analyse nous fixons[1] δD=1. Pour le reste le principe est identique, c.à.d. avec une augmentation vers la droite de la figure de l'unique échéance relative alors que les périodes restent constantes. Durant l'intervalle $[-70, 70]$, certains couples *(T,D)* ont leurs échéances relatives supérieures aux périodes alors que les autres ont leurs échéances relatives inférieures aux périodes. $x < -70$ (respectivement $x > -70$) signifie que toutes les échéances relatives sont inférieures (respectivement supérieures) aux périodes. Nous remarquons que :

- lorsque $x \geq 0$, $\varepsilon(RM)$ augmente avec x, c.à.d. la valeur de l'unique échéance relative. Ceci est encore une fois le comportement prévu sur la minoration de $\varepsilon(P)$ (cf. chapitre VI.1.3.2.b p. 57). Notons toutefois que comme $x = 0$ ne représente pas le cas $\forall i, D_i \geq T_i$, nous n'avons pas immédiatement $N_{EDF}(\tau) = N_U(\tau) = N'_U(\tau)$.
- durant l'intervalle [-50, 40], $\varepsilon(RM)$ est à son plus bas, sous la borne de Liu & Layland. Ceci confirme l'avantage en terme d'efficacité de *EDF* sur les algorithmes à priorités fixes lorsque certains couples *(T,D)* ont leurs échéances relatives supérieures aux périodes alors que les autres ont leurs échéances relatives inférieures aux périodes.
- lorsque $x \leq -50$, $\varepsilon(RM)$ tend rapidement vers 1 lorsque x diminue. Ceci n'est pas reflété par le théorème d'efficacité qui donne alors une forte minoration. Notons cependant que la minoration se stabilise à nouveau dès que toutes les échéances relatives deviennent inférieures aux périodes (ici à partir de x=-70) grâce à l'utilisation de la norme N'_U au lieu N_U.

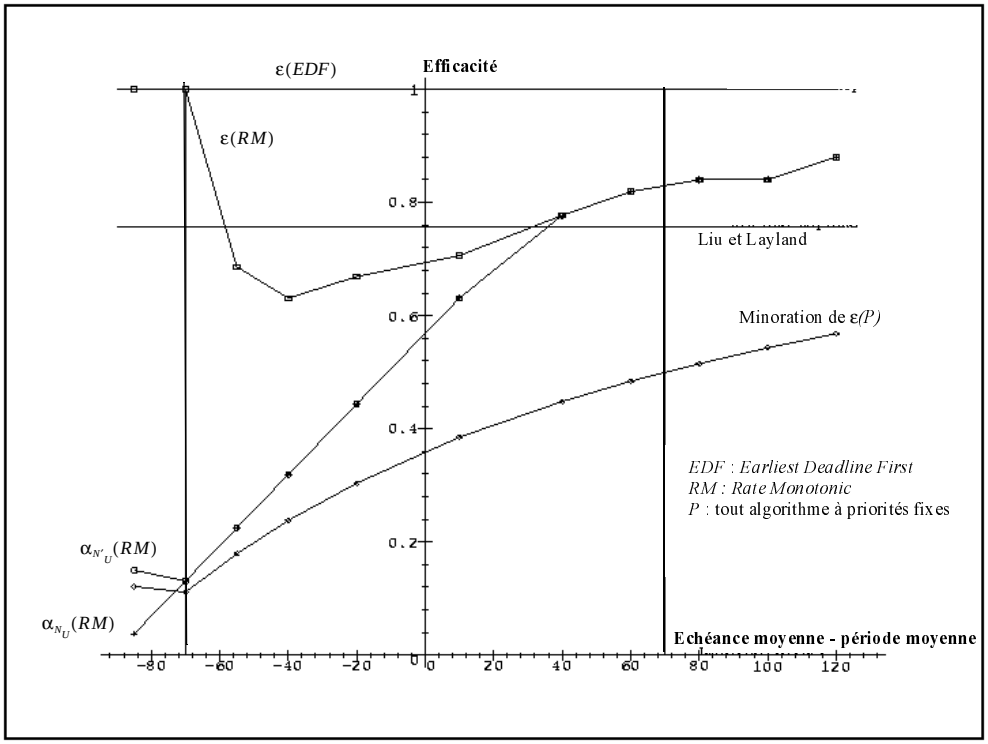
Pour conclure, nous avons bien vérifié que l'efficacité de tout algorithme d'ordonnancement à priorités fixes ne peut s'annuler et se rapproche même de l'efficacité de *EDF* lorsque les trafics sont homogènes et/ou présentent des échéances relatives supérieures aux périodes. Les limitations de nos mesures portent sur :

- le coût du calcul exact de l'efficacité par N_{EDF} qui ne nous a pas permis des expérimentations sur des référentiels présentant un grand nombre de couples *(T,D)*.
- les théorèmes d'efficacité qui conduisent à de fortes approximations ne permettant pas une interprétations claire de tous les cas.

Notre opinion est cependant que ces deux points sont perfectibles. Des informations complémentaires pourront d'ailleurs être trouvées dans [Hermant 1998] et [Leboucher 1998].

1. Nous utilisons alors *RM* à la place de *DM* car ce dernier algorithme ne permet pas d'attribuer les priorités de façon pertinente lorsque δD=1. De même, nous n'utilisons pas le deuxième théorème d'efficacité spécialisé pour *DM*.

Figure 22 : Influence des échéances avec δD=1



X.2. Le cas non-préemptif non-oisif

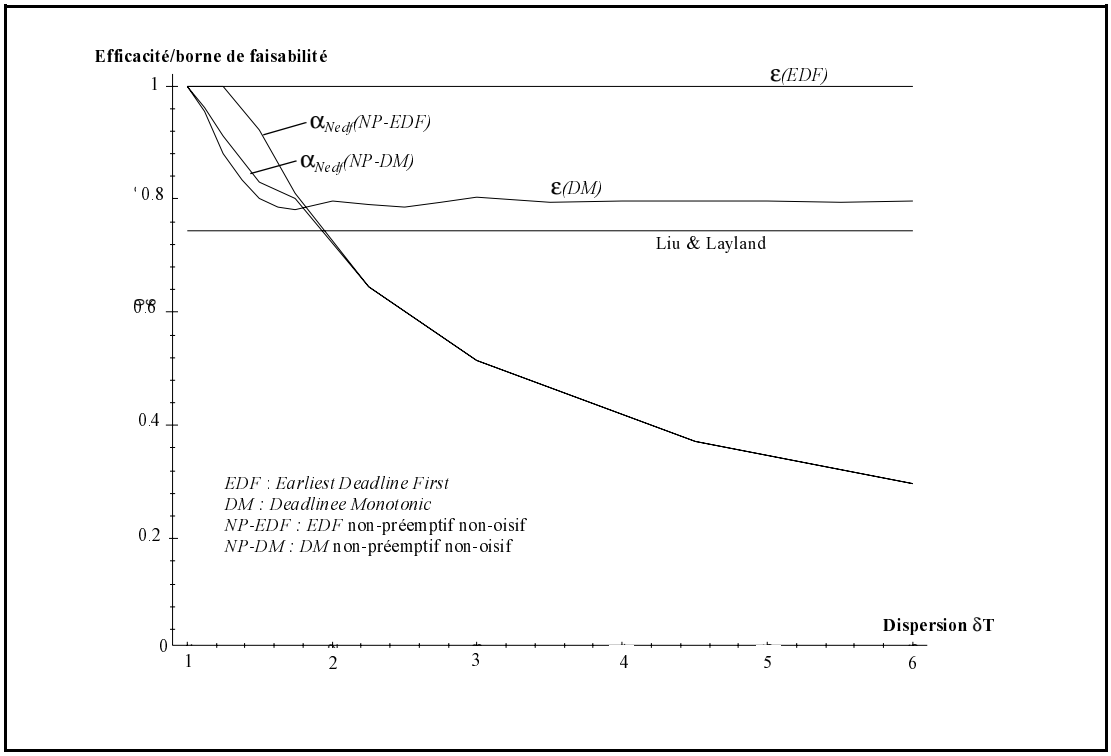
L'objectif est d'illustrer les tendances observées lors du chapitre IX.1. p. 85, sur l'influence de l'absence de préemption en terme d'efficacité. Plus précisément nous reprenons les exemples précédents en y ajoutant $\alpha_{N_{EDF}}(NP\text{-}EDF)$ et $\alpha_{N_{EDF}}(NP\text{-}DM)$, les bornes de faisabilité exactes obtenues sur les versions non-préemptives de *EDF* et *DM*. Rappelons que nous ne pouvons pas parler d'efficacité pour *NP-EDF* et *NP-DM* car nous n'avons pas prouvé que N_{EDF} est un critère d'optimalité.

X.2.1. Influence de la dispersion

Commençons par reprendre l'exemple de la Figure 19 p. 95, avec $\forall i, D_i = T_i$ (le cas de Liu & Layland) et la dispersion $\delta T = Max(T)/Min(T)$ qui augmente en abscisse (cf. Figure 23). Comme attendue (cf. chapitre IX.1. p. 85), lorsque la dispersion :

- augmente, l'intérêt de *NP-EDF* face à *NP-DM* disparaît. Ceci est contraire au comportement de *EDF* face à *DM* en préemptif.
- est faible, nous observons que $\alpha_{N_{EDF}}(NP\text{-}DM)$ peut être supérieur à ε(*DM*), l'efficacité de *DM*. Bien que ce phénomène soit limité, ceci confirme bien qu'un algorithme d'ordonnancement à priorités fixes préemptif ne domine[1] pas nécessairement son homologue non-préemptif non-oisif (cf. chapitre VIII.2.3. p. 82).

Figure 23 : Influence de la dispersion dans le cas de Liu et Layland avec 5 couples *(T,D)*



1. au sens de la Définition 35 p. 48.

X.2.2. Influence des échéances relatives

La Figure 24 s'inspire de la Figure 20 p. 97. Vers la gauche les échéances relatives diminuent en comparaison des périodes, vers la droite elles augmentent ($x = 0 \Rightarrow \forall i, D_i = T_i$, $x < 0 \Rightarrow \forall i, D_i < T_i$ et $x > 0 \Rightarrow \forall i, D_i > T_i$). Les valeurs prises ici sont cependant différentes car on impose une translation des échéances. $\delta D = Max(D_i)/Min(D_i)$, la dispersion en échéance, est donc bien plus forte vers la gauche. Les tendances du chapitre IX.1. p. 85, sont confirmées :

- dans le cas $\forall i, D_i > T_i$, l'influence de l'absence de préemption diminue en allant vers la droite de la figure (c.à.d. lorsque δD diminue et que les échéances augmentent face aux périodes). On constate en effet que les bornes de faisabilité des algorithmes non-préemptifs non-oisifs tendent vers celles de leurs homologues préemptifs. Notons que $\alpha_{N_{EDF}}(NP\text{-}EDF)$ tend rapidement vers 1 alors que $\alpha_{N_{EDF}}(NP\text{-}DM)$ s'aligne sur $\varepsilon(DM)$, c.à.d. tend lentement vers 1 (cf. chapitre VI.1.4. p. 58).
- dans le cas $\forall i, D_i < T_i$, l'influence de l'absence de préemption est très nette lorsque l'on va vers la gauche de la figure (c.à.d. lorsque δD augmente et que les échéances diminuent face aux périodes). $\alpha_{N_{EDF}}(NP\text{-}EDF)$ et $\alpha_{N_{EDF}}(NP\text{-}DM)$ se rejoignent et diminuent rapidement. Encore une fois l'intérêt de *NP-EDF* par rapport à un algorithme non-préemptif non-oisif à priorités fixes disparait rapidement dans ce cas.

NP-EDF est optimal parmi les algorithmes non-préemptifs non-oisifs en présence de trafics non-concrets (cf. Théorème 23 p. 77). Les mesures précédentes nous montrent cependant que, contrairement au cas préemptif, ceci ne se traduit pas un avantage net en terme de borne de faisabilité. Afin de mieux situer les référentiels d'ordonnancement où *NP-EDF* peut être intéressant, la Figure 25 reprend la figure Figure 22 p. 99, en y ajoutant la mesure de $\alpha_{N_{EDF}}(NP\text{-}EDF)$ et de $\alpha_{N_{EDF}}(NP\text{-}RM)$. Rappelons qu'il s'agit d'un cas général où à aucun moment nous n'avons $\forall i, D_i = T_i$, mais où δD, la dispersion en échéance vaut 1. Nous remarquons que dans ce cas les bornes de faisabilité des algorithmes non-préemptifs non-oisifs sont proches de leurs homologues préemptifs. Plus précisement :

- $\alpha_{N_{EDF}}(NP\text{-}EDF) = \varepsilon(EDF) = 1$, ce qui était prévu par le chapitre IX.1.2. p. 86.
- $\alpha_{N_{EDF}}(NP\text{-}RM)$ est même supérieur à $\varepsilon(RM)$, notamment lorsque l'échéance unique est légèrement supérieure à la période moyenne. Ceci confirme à nouveau la non dominance[1] d'un algorithme d'ordonnancement à priorités fixes préemptif face à son homologue non-préemptif non-oisif (cf. chapitre VIII.2.3. p. 82).

Il apparaît que *NP-EDF* reste plus intéressant que tout autre algorithme d'ordonnancement non-préemptifs non-oisifs à priorités fixes, principalement lorsque la dispersion en échéance est faible et que certaines tâches ont leurs échéances relatives supérieures aux périodes alors que les autres ont leurs échéances relatives inférieures aux périodes.

1. au sens de la Définition 35 p. 48.

Figure 24 : Influence des échéances et de leur dispersion

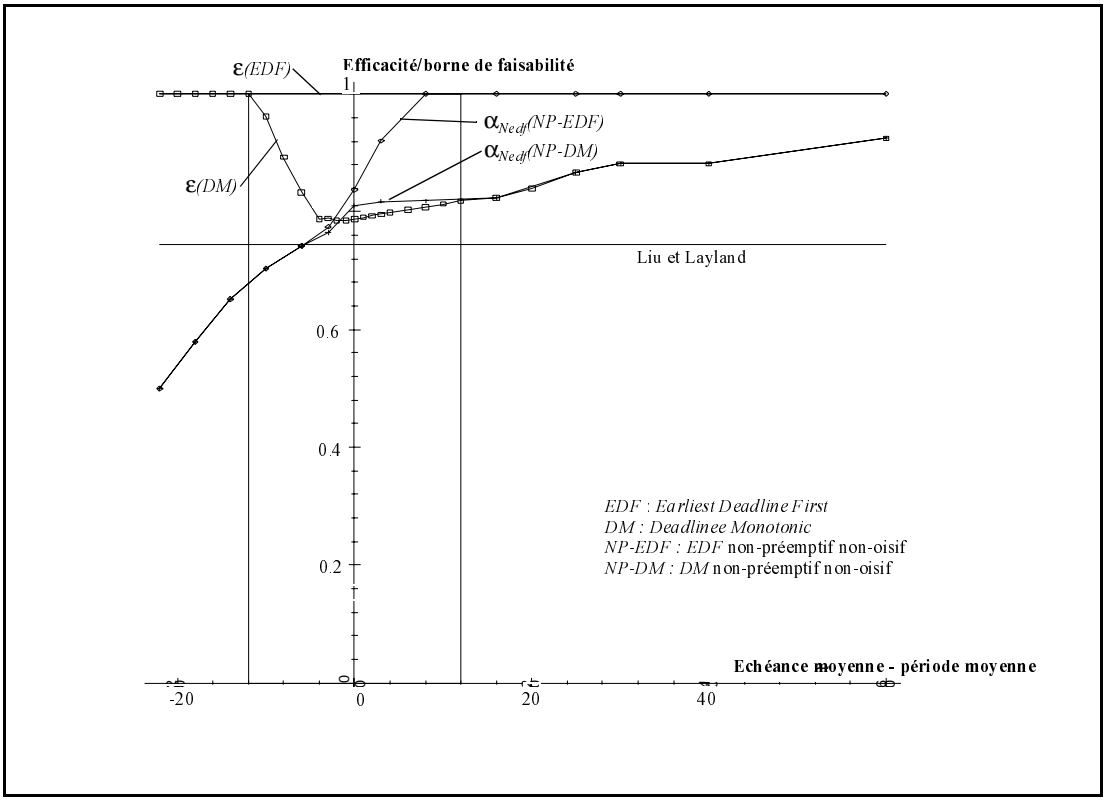
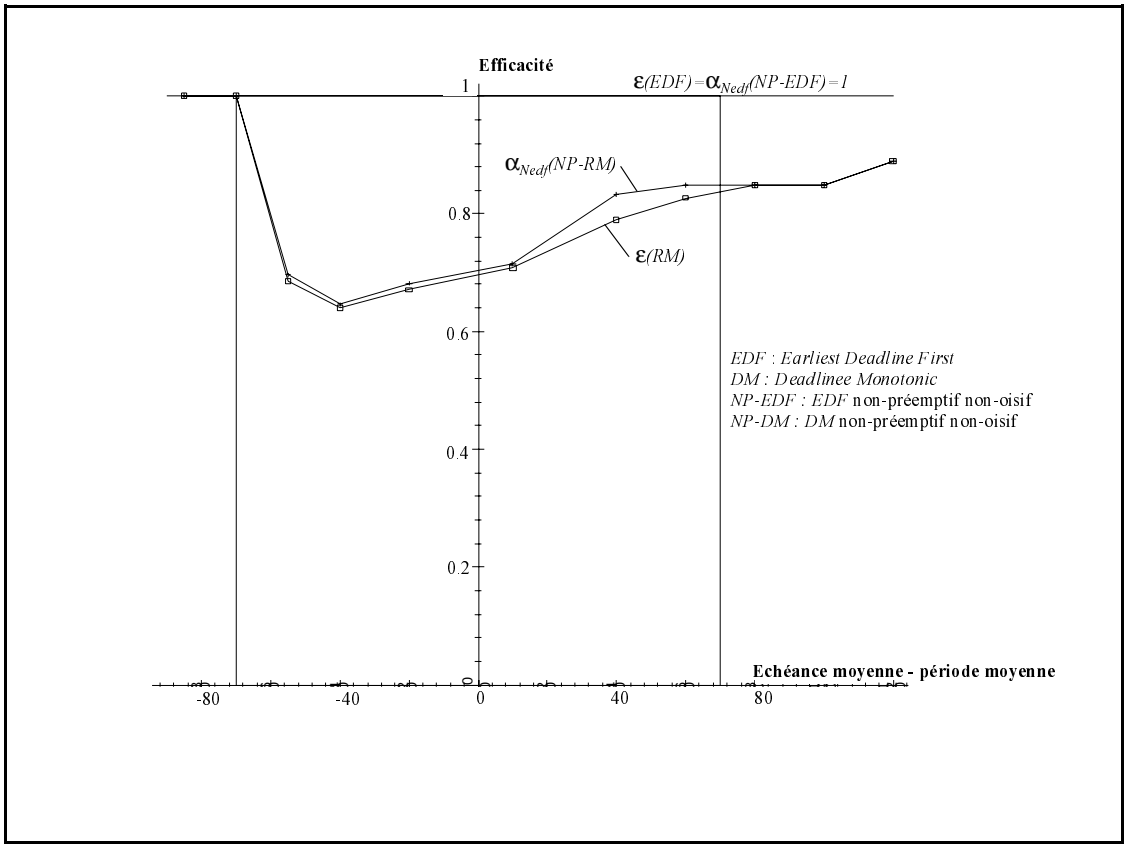


Figure 25 : Influence des échéances avec δD=1



XI. Synthèse

Notre objectif était d'identifier, étendre et comparer quelque peu les outils et solutions à un problème d'ordonnancement temps réel générique (cf. chapitre I. p. 1). Les référentiels d'ordonnancement $\Sigma=(\Upsilon,\Pi)$ (cf. Définition 21 p. 20) nous ont permis de poser :

- Υ, le problème temps réel modélisé par un ensemble de trafics non-concrets τ (cf. Hypothèse 1 p. 9), encore appelés TPN (cf. définition 27 p. 22), à ordonnancer sur un serveur. Nous avons retenu ces trafics car ils n'imposent ni connaissances sur les instants d'activation des tâches, ni relations particulières entre les périodes et les échéances dans le cas général. Tout au plus doit-on respecter une densité maximale d'activation par tâche.
- Π, l'ensemble d'algorithmes candidats à résoudre ce problème. Nous avons retenu des algorithmes en-ligne, non-probabilistes, préemptifs/non-préemptifs non-oisifs (cf. Hypothèse 2 p. 10) et les priorités fixes/dynamiques. Les preuves d'optimalité couvrent d'autres gestions de priorités possibles (*FIFO*, *round robin*...) non traités ici.
- les concepts de faisabilité (cf. Définition 23 p. 21), d'optimalité (cf. Définition 24 p. 21) et d'efficacité (cf. Définition 38 p. 53) utilisés.
- les énoncés des résultats d'ordonnancement sans ambiguïtés.

Nous avons fixé dans notre cahier des charges[1] (cf. chapitre II.2. p. 8) les hypothèses et les propriétés "temps réel" que nous voulions vérifier. Les résultats énoncés sont issus de l'état de l'art, des deux rapports co-signés [George et al. 1996] et [Hermant et al. 1996] et de cette thèse. Nous récapitulons tout d'abord, dans les deux tables suivantes, ces résultats en faisant varier Υ et Π. Nous les commentons ensuite en reprenant chacune des propriétés et en précisant notre contribution.

Table 5 : Ordonnancement préemptif

	EDF	**Priorités fixes**
Faisabilité seule du trafic τ : - Propriété 1 p. 8 (test) - Propriété 4 p. 12 (**coût**)	- Théorème 12 p. 46. - chapitre VI.2.2.3. p. 62.($O(n^2)$)	
Faisabilité et pires temps de réponse des tâches - Propriété 2 p. 11 (test) - Propriété 4 p. 12 (**coût**)	- Théorème 13 p. 47. - chapitre VI.2.2.5. p. 63 ($O(n^3)$)	- Théorème 14 p. 50. - chapitre VI.2.2.4. p. 63 ($O(n^3)$)
Optimalité des algorithmes (Propriété 3 p. 11)	- Théorème 1 p. 29. - Théorème 12 p. 46 (par mesure de la demande processeur)	- Théorème 10 p. 40 (*Audsley* pour trafics généraux) - Théorème 7 p. 37. (*DM* si $T_i \geq D_i$, $\forall i \in [1,n]$) - Théorème 5 p. 36. (*RM* si $T_i = D_i$, $\forall i \in [1,n]$)
Efficacité des algorithmes (Propriété 3 p. 11)	Théorème 17 p. 54 ($\varepsilon(EDF) = 1$)	- Théorème 18 p. 56. - Théorème 19 p. 59 (avec *DM*)

1. De nombreuses modifications de ce cahier des charges sont possibles. A titre d'exemple, le lecteur intéressé trouvera quelques éléments concernant la gigue d'activation et les facteurs de blocage en annexe B, p. 121.

Table 6 : Ordonnancement non-préemptif non-oisif

	NP-EDF	Priorités fixes
Faisabilité seule du trafic τ : - Propriété 1 p. 8 (Calcul) - Propriété 4 p. 12 (**coût**)	- Théorème 23 p. 77. - chapitre IX.2.1. p. 89 ($O(n^2)$)	
Faisabilité et pires temps de réponse des tâches - Propriété 2 p. 11 (Calcul) - Propriété 4 p. 12 (**coût**)	- Théorème 24 p. 78. - chapitre IX.2.3. p. 90 ($O(n^3)$)	- Théorème 25 p. 81. - chapitre IX.2.2. p. 89 ($O(n^3)$)
Optimalité des algorithmes (Propriété 3 p. 11)	Théorème 23 p. 77 (par mesure de la demande processeur uniquement).	Théorème 26 p. 83. (*Audsley* uniquement)
Efficacité/borne de faisabilité (Propriété 3 p. 11)	chapitre IX.1.2. p. 86. (pas de minoration)	chapitre IX.1.3. p. 87. (pas de minoration)

XI.1. Faisabilité des trafics et pires temps de réponse des tâches

La définition choisie pour la **faisabilité** énonce qu'un trafic non-concret $\tau \in \Upsilon$ est faisable dans Σ si un même algorithme $Q \in \Pi$ peut ordonnancer toute instanciation concrète ω de τ (cf. Définition 23 p. 21). Notons que :

- d'autres définitions possibles de la faisabilité ont été discutées lors du chapitre III.3.2. p. 22. Nous avons signalé que le choix de la "bonne" définition ne dépendait pas tant du "fournisseur" de solutions algorithmiques que du problème posé par le client.
- la faisabilité dans Σ peut être établie seule (cf. Propriété 1 p. 11) ou grâce au calcul des **pires temps de réponse** des tâches (cf. Propriété 2 p. 11).

Les résultats pour ces deux propriétés sont résumés dans les tables précédentes. Nous rappelons la démarche utilisée pour les établir dans le cas général en insistant sur les périodes occupées de priorités maximales. Dans tous les cas (préemptif/non-préemptif, priorités fixes/ dynamiques), l'intervalle d'étude pour la faisabilité est borné par L, taille de la période occupée synchrone du processeur (cf. chapitre III.2.1. p. 17), et l'interarrivée des tâches est périodique.

Avec les priorités dynamiques (cf. Table 7 p. 105) :

- nous avons considéré l'algorithme *EDF* en préemptif (cf. chapitre IV.2.1. p. 28) et son homologue *NP-EDF* en non-préemptif non-oisif (cf. chapitre VII.2. p. 69). Nous avons introduit le concept de *deadline busy period* (cf. Définition 19 p. 19), symétrique de celui de *level-i busy period* utilisé avec les priorités fixes, qui nous a permis :
  - d'identifier les pires scénarii d'activation avec, pour *NP-EDF*, la pire inversion de priorités ($B(t)=max_{D_i>t}\{C_i-1\}$ qui disparaît donc au bout de $t > max\{D_i\}$),
  - de proposer, par tâche τ_i de τ, les bornes $L_i^S \le L_i \le L$ sur les intervalles d'études.
- pour la faisabilité d'un trafic non-concret τ, le scénario synchrone (cf. Définition 11 p. 15) est le pire cas avec *EDF* (cf. Lemme 5 p. 30). La formulation du test nécessaire et suffisant (cf. Théorème 12 p. 46) est en $\forall t \in S, P(t)$, où P est un prédicat basé sur la demande processeur $h(t)$ (cf. définition 15 p. 16) et S un ensemble borné de points où tester ce prédicat. Notons que *NP-EDF* ne fait qu'introduire le terme additionnel $B(t)$ dans le prédicat (cf. Théorème 23 p. 77), un trafic faisable par *NP-EDF* est donc faisable par *EDF*. Nous avons vu deux approches pour borner l'intervalle d'étude (manipulations algèbriques de la demande processeur et périodes occupées). En particulier, les *deadline busy periods* nous ont permis de proposer, par tâche τ_i de τ, la borne L_i^S comme un cas particulier de la borne L_i utilisée pour le calcul des pires temps de réponse de τ_i.

• pour calculer le pire temps de réponse de toute tâche τ_i de τ, les scénarii d'activation pires cas sont déduits du scénario synchrone par décalage des activations de τ_i (cf. chapitre IV.2.4. p. 33). Chacun des scénarii d'activation obtenu conduit alors à calculer l'instant de fin d'exécution de l'activation de τ_i arrivée en a par un calcul de période occupée $L_i(a)$. Le test résultant est en $\forall i \in [1, n], r_i \leq D_i$ avec $r_i = max_a\{L_i(a) - a\}$. Les *deadline busy periods* nous ont permis d'établir le calcul des pires temps de réponse en non-préemptif non-oisif (cf. Théorème 24 p. 78) et de constater que ceux-ci n'étaient pas nécessairement supérieurs à ceux du cas préemptif (cf. Théorème 13 p. 47). En effet, bien que les inversions de priorités tendent à augmenter les temps de réponse, le calcul non-préemptif non-oisif de $L_i(a)$ s'arrête avant l'exécution de l'activation de la tâche τ_i activée en a, contrairement au cas préemptif (cf. chapitre VIII.1.2.2. p. 75). Les *deadline busy periods* nous ont aussi permis d'établir, par tâche τ_i, la borne $L_i \leq L$ sur l'intervalle d'étude comme la valeur maximale de $L_i(a)$ vérifiant $L_i(a) > a$ (cf. chapitre V.1.1.2. p. 43).

Table 7 : Utilisation des *deadline busy periods*

	EDF	***NP-EDF***
Deadline busy periods pour la faisabilité de τ	Lemme 9 p. 41.	Lemme 17 p. 73.
Deadline-d busy periods pour les pires temps de réponse des tâches.	Lemme 10 p. 42.	Lemme 18 p. 74.
$L_i(a)$	Equation (17) p. 46.	Equation (33) p. 78.
$L_i^s \leq L_i \leq L$	chapitre V.1.1.2. p. 43.	chapitre VIII.1.2.2. p. 75.

Avec les priorités fixes (cf. Table 8) :

• les *level-i busy periods* permettent d'identifier les scénarii d'activation pires cas, de borner les intervalles d'étude et, en non-préemptif non-oisif, de mesurer la pire inversion de priorité possible pour la tâche τ_i de τ ($B_i = max_{k \in lp(i)}\{C_k - 1\}$, qui ne disparaît donc pas au bout de $t > max\{D_i\}$ comme pour *NP-EDF*).

• nous ne connaissons pas de formulation qui ne fasse pas appel (même implicitement) aux calculs des pires temps de réponse pour établir la faisabilité d'un trafic non-concret τ.

• pour calculer le pire temps de réponse de toute tâche τ_i de τ, contrairement à *EDF*, seul le scénario synchrone est pertinent en préemptif (cf. Lemme 8 p. 38). Il conduit à tester chaque $(q+1)^{ième}$ activation de τ_i arrivée en qT_i par un calcul de période occupée $w_{i,q}$ (cf. Equation (18) p. 49). Le test est en $\forall i \in [1, n], r_i \leq D_i$ avec $r_i = max_q\{w_{i,q} - qT_i\}$. Notons encore une fois que les pires temps de réponse du cas non-préemptif non-oisif ne sont pas nécessairement supérieurs à ceux du cas préemptif. En effet, bien que les inversions de priorités tendent à augmenter les temps de réponse, le calcul non-préemptif non-oisif de $w_{i,q}$ s'arrête avant l'exécution de l'activation de la tâche τ_i activée en qT_i, contrairement au cas préemptif (cf. chapitre VIII.2.1.2. p. 80). Le seul test dont nous disposons avec les priorités fixes ne permet donc pas de dire qu'un trafic faisable en non-préemptif non-oisif est faisable avec la version préemptive du même algorithme à priorités fixes. La borne obtenue pour calculer le pire temps de réponse par tâche τ_i vaut $L_i \leq L$.

Table 8 : Utilisation des *level-i busy periods*

	***P*, un algorithme à priorités fixes**	***NP-P*, sa version non-préemptive**
level-i busy periods pour la faisabilité de τ et les pires temps de réponse des tâches.	Lemme 8 p. 38.	Lemme 21 p. 79.
$w_{i,q}$	Equation (15) p. 39.	Equation (37) p. 81.
L_i	Lemme 15 p. 50.	Equation (35) p. 80.

Ces résultats étaient connus dans l'ensemble. Dans le cas préemptif, nous avons principalement introduit les *deadline busy periods,* symétriques aux *level-i busy periods* utilisées avec les priorités fixes, pour améliorer les bornes, par tâche τ_i de τ, sur les intervalles d'études avec *EDF*. Dans le cas non-préemptif non-oisif, symétrique avec le cas préemptif, nous avons établi les calculs de pires temps de réponse avec *NP-EDF* en nous appuyant sur ces *deadline busy periods*. Les différences avec le cas préemptif portent sur les scénarii pires cas, les inversions de priorités et les calculs de $L_i(a)$. Une question ouverte subsiste : existe-il un calcul de bornes à l'aide des *deadline busy periods* moins coûteux que celui proposé, ou est-il possible de l'effectuer en-ligne comme avec les priorités fixes (cf. chapitre IV.3.4.2. p. 39) ?

XI.2. Optimalité et efficacité des algorithmes

Le danger est d'interpréter ces propriétés hors de leur contexte. Les référentiels d'ordonnancement Σ permettent de fixer les ensembles de trafics Υ et d'algorithmes Π auxquels se réfère la validité d'un résultat d'ordonnancement. De même, la définition de la faisabilité dans Σ (cf. Définition 23 p. 21) est fondamentale puisque ces propriétés y font référence. Nous précisons les résultats d'optimalité et d'efficacité obtenus en jouant sur les couples *(T,D)* (cf. Définition 27 p. 22) représentés dans Υ et les algorithmes retenus dans Π.

XI.2.1. Optimalité

La définition choisie énonce qu'un algorithme Σ-optimal (cf. Propriété 3 p. 11) sait ordonnancer tout trafic non-concret τ Σ-faisable, c.à.d. tel que toutes les instanciations concrètes ω de τ soient ordonnançables par un même algorithme dans Σ (cf. Définition 24 p. 21). La Σ-optimalité est une propriété séduisante mais difficile à établir et/ou à comparer. Nous avons introduit pour cela la propriété de Υ-dominance (cf. définition 35 p. 48) qui affaiblit la Σ-optimalité en énonçant qu'un algorithme en domine un autre si les trafics faisables dans Σ par le deuxième le sont aussi par le premier. Nous donnons en fin de ce chapitre quelques relations d'ordre classiques sur la Υ-dominance, ainsi que quelques impossibilités pratiques.

Table 9 : Σ-optimalité

Υ / Π	Trafics non-concrets τ $T_i = D_i, \forall i \in [1,n]$	Trafics non-concrets τ $T_i \geq D_i, \forall i \in [1,n]$	Trafics non-concret τ cas général
Tout algorithme	*EDF*	*EDF*	*EDF*
Tout algorithme non-préemptif non-oisif	*NP-EDF*	*NP-EDF*	*NP-EDF*
Algorithmes préemptifs à priorités fixes	*RM=DM*	*DM*	[*Audsley*]
Algorithmes non-préemptifs non-oisifs à priorités fixes	~~*RM*~~, [*Audsley*]	~~*DM*~~, [*Audsley*]	[*Audsley*]
Algorithmes à priorités fixes	~~*RM*~~, [*Audsley*]	~~*DM*~~, [*Audsley*]	[*Audsley*]

Nous attirons l'attention du lecteur sur le fait que cette table[1] ne couvre qu'un nombre limité de problèmes d'ordonnancement centralisé basés sur notre définition de la Σ-optimalité et notre modèle de trafics non-concrets τ. Même en se limitant à ce contexte, il serait possible de considérer d'autres cas particuliers sur ϒ et Π. Cette table donne toutefois un certain nombre d'informations connus dans l'ensemble mais que nous relativisons.

EDF est Σ-optimal quels que soient les trafics non-concrets τ considérés dans ϒ et quels que soient les algorithmes préemptifs/non-préemptifs, oisifs/non-oisifs, à priorités fixes/ dynamiques dans Π. Ce résultat est issu d'une condition nécessaire pour tout algorithme (basée sur la demande processeur et le scénario synchrone) qui est aussi suffisante pour *EDF* (cf. Théorème 12 p. 46). Il est même plus général car une autre preuve (cf. Théorème 1 p. 29) montre que tout ordonnancement valide peut être ramené par permutation à un ordonnancement *EDF* valide. Elle s'applique donc à tout trafic concret ω dans ϒ (cf. chapitre III.3.2. p. 22, pour une discussion de cette définition de la Σ-optimalité).

Dans le cas non-préemptif non-oisif, la Σ-optimalité de *NP-EDF* s'appuie à nouveau sur la demande processeur (cf. Théorème 23 p. 77) mais le raisonnement par permutation ne tient plus (cf. chapitre VIII.1.1. p. 72). Ce résultat est à relativiser puisqu'il est valable uniquement pour des trafics non-concrets τ et pour des algorithmes non-préemptifs non-oisifs dans Π. Le parallèle avec *EDF* est ainsi limité aussi bien sur ϒ que sur Π. Si pour des raisons que nous ne discuterons pas ici le cahier des charges impose un tel référentiel, alors la Σ-optimalité de *NP-EDF* ne peut cependant être mise en défaut.

si Π ne contient que des algorithmes préemptifs à priorités fixes et ϒ des trafics non-concrets τ, alors la procédure d'*Audsley* est Σ-optimale dans le cas général (c.à.d. quelles que soient les périodes et échéances des tâches), *DM* est Σ-optimal lorsque $D_i \leq T_i, \forall i \in [1, n]$ (cf. Théorème 7 p. 37) et *RM* est Σ-optimal lorsque $D_i = T_i, \forall i \in [1, n]$ (cf. Théorème 5 p. 36). Notons cependant que dans le cas général : d'une part, la procédure d'*Audsley* n'est pas totalement satisfaisante car elle améliore en fait en $O(n^2)$ une procédure Σ-optimale en $n!$ (avec n le nombre de tâches) qui testerait toutes les permutations possibles pour assigner les priorités (cf. Théorème 10 p. 40). Elle reste donc coûteuse à mettre en oeuvre et dominée[2] par *EDF* ; d'autre part, il n'y a pas de relation de dominance entre *DM* ou *RM* (cf. chapitre V.2.1. p. 47). Une question ouverte est donc de trouver un algorithme préemptif à priorités fixes Σ-optimal et à faible coût dans le cas général (ou même par défaut ϒ-dominant).

si Π ne contient que des algorithmes non-préemptifs non-oisifs à priorités fixes et ϒ des trafics non-concrets τ, alors la procédure d'*Audsley* reste Σ-optimale dans le cas général (cf. Théorème 26 p. 83). Par contre, contrairement au cas préemptif, il n'y a pas de procédure de repli pour attribuer les priorités à faible coût car nous ne connaissons pas d'algorithme Σ-optimal, ni même ϒ-dominant, dans des référentiels plus restreints (cf. chapitre VIII.2.3. p. 82)

Notons que la dernière ligne de cette table, qui mixe dans Π les algorithmes à priorités fixes préemptifs et non-préemptifs non-oisifs, présente le même problème. Autrement dit les résultats d'optimalité sur *DM* et *RM* ne tiennent plus dès que Σ contient des algorithmes non préemptifs non-oisifs (cf. chapitre VIII.2.3. p. 82). La faisabilité des trafics non-concrets τ est en effet basée uniquement sur des calculs de pires temps de réponse dont les calculs sont différents en préemptif et en non-préemptif non-oisif.

1. les noms rayés signalent des algorithmes non optimaux dans le référentiel considéré (cf chapitre VIII.2.3. p. 82), contrairement à ce que pourrait suggérer le cas préemptif. Les cases grisées signalent des référentiels sans résultats d'optimalité dans la littérature (à notre connaissance et pour notre modèle de trafic pourtant générique).
2. toujours au sens de la Définition 35 p. 48.

Dans le cas général, et contrairement à *EDF* et *NP-EDF* avec la demande processeur, il n'existe pas de condition de faisabilité nécessaire pour un sous ensemble d'algorithmes à priorités fixes, qui soit aussi suffisante pour l'un d'entre eux. Une question ouverte est donc de trouver de telles conditions qui seraient des pistes pour d'éventuels résultats d'optimalité ou de dominance dans des référentiels d'ordonnancement limités.

Sans apporter de nouveautés en termes d'optimalité des algorithmes nous avons cherché à illustrer la difficulté à atteindre et/ou à comparer cette propriété. Ainsi, pour les trafics concrets ω, seul l'algorithme *EDF* a été montré Σ-optimal. Pour les trafics non-concrets τ qui nous intéressent, la Σ-optimalité est relaxée (cf. chapitre III.3.2. p. 22, pour une discussion) et des résultats existent[1]. Ceux-ci restent toutefois assez sensibles, en particulier avec les priorités fixes où ils ne résistent pas aux référentiels d'ordonnancement contenant des algorithmes non-préemptifs (excepté la procédure d'*Audsley*, coûteuse et dominée par *NP-EDF*).

Pour conclure, récapitulons ce que nous connaissons grâce à la propriété de Υ-dominance[2] (notée > ici). Celle-ci permet de comparer quelque peu les résultats de Σ-optimalité grâce à des relations d'ordre (cf. Table 10 où *P* est un algorithme à priorités fixes quelconque). Ces relations d'ordre traduisent en fait des référentiels de plus en plus réduits où s'appliquent les résultats de Σ-optimalité :

Table 10 : Υ-dominance

Préemptif	*EDF > Audsley > DM* *DM > RM uniquement si* $D_i \leq T_i$, $\forall i \in [1, n]$
Non-préemptif non-oisif	*NP-EDF > Audsley > NP-P*
Préemptif et Non-préemptif non-oisif	*EDF > NP-EDF* *par contre P > NP-P ou NP-P > P sont faux*

XI.2.2. Efficacité

Afin de mesurer un peu mieux les performances des algorithmes, nous nous sommes ensuite intéressé à $\varepsilon(P)$, l'efficacité (cf. Propriété 3 p. 11) de tout algorithme *P* dans un référentiel d'ordonnancement Σ donné qui contient *EDF* et des algorithmes à priorités fixes.

En préemptif, L. Leboucher formalise $\varepsilon(P)$ dans [Hermant et al. 1996], comme une mesure de la proximité à l'optimalité de *EDF*. Une relation d'équivalence en périodes et en échéances (cf. Définition 36 p. 52) pour la faisabilité des trafics non-concrets τ permet d'utiliser la demande processeur pour établir N_{EDF}, la meilleure norme possible dans l'espace vectoriel $(\Upsilon, +, \cdot)$ pour cette mesure (cf. Théorème 17 p. 54). *EDF* étant l'algorithme le plus efficace ($\varepsilon(EDF) = 1$), L. Leboucher propose alors une minoration de l'efficacité de tout algorithme à priorités fixes et une interprétation de cette minoration en fonction des couples *(T,D)* (cf. Définition 27 p. 22) formant la base de l'espace vectoriel $(\Upsilon, +, \cdot)$. Ce résultat généralise l'approche de [Liu and Layland 1973] dans le cas $D_i = T_i$, $\forall i \in [1, n]$.

Nous avons spécialisé cette minoration pour $\varepsilon(DM)$ (cf. Théorème 19 p. 59) car nous ne connaissons pas d'algorithme à priorités fixes optimal ou dominant à faible coût dans un référentiel d'ordonnancement plus général (cf. chapitre V.2.1. p. 47). Les résultats obtenus

1. *RM, DM, Audsley* pour le cas préemptif à priorités fixes ou *EDF* et *Audsley* dans leurs versions non-préemptives non-oisives. Notons que nous avons exclu les algorithmes non-préemptifs oisifs qui, bien que théoriquement supérieurs, ne cadrent pas avec notre cahier des charges car ils imposent une hypothèse de clairvoyance sur le futur pour prendre les décisions d'ordonnancement valides (cf. chapitre VII.1.1. p. 68).
2. au sens de la Définition 35 p. 48.

pour *DM* confirment les tendances observées dans [Hermant et al. 1996], où il apparaît que l'efficacité de tout algorithme à priorités fixes ne peut s'annuler et se rapproche d'autant de l'efficacité de *EDF* que les couples *(T,D)* dans Σ sont homogènes (c.à.d. que les dispersions en périodes δT et en échéances δD tendent vers 1) et/ou que les échéances relatives sont grandes face aux périodes (cf. chapitre VI.1.3.2. p. 55). Par exemple pour des trafics multimédia.

Nous avons complété ces informations lorsque l'algorithme à priorités fixes utilisé est *DM* (cf. chapitre VI.1.4. p. 58) :

- si $D \geq T$ pour tout couple *(T,D)* de Υ, alors $\varepsilon(DM)$ est minoré par 1/2 quel que soit le nombre de couples *(T,D)* formant la base de l'espace vectoriel $(\Upsilon, +, \cdot)$. Ce résultat peut être comparé à la borne $n(2^{1/n} - 1)$ obtenue par [Liu and Layland 1973] dans le cas $\forall i, D_i = T_i$ avec *RM*. Précisons que l'augmentation de l'efficacité de *DM*, lorsque les échéances augmentent, se produit ici dès lors que $D_i \geq min_{j \leq i}(T_j)$ pour toute tâche τ_i de τ, et tend toujours vers 1 lorsque $\{D_i\} \gg \{T_i\}$ (c.à.d. lorsque $D \geq max(T)$ pour tout couple *(T,D)* de Υ).
- si $\{D_i\} \ll \{T_i\}$ (c.à.d. lorsque $D \leq min(T)$ pour tout couple *(T,D)* de Υ) la minoration de $\varepsilon(DM)$ tend vers $1/\delta T$ lorsque $max(D)$ devient négligeable devant $max(T)$. Elle tend donc vers 1 lorsque la dispersion en périodes δT diminue aussi. L'intuition est ainsi confirmée puisque ces deux facteurs conduisent à une période occupée synchrone (cf. définition 17 p. 17), la plus contrainte en terme de faisabilité en préemptif, n'ayant qu'une activation par tâche. Elle est donc ordonnancée de la même façon par *EDF* et *DM*.

Du point de vue de l'efficacité, il est donc d'autant plus pertinent d'utiliser *DM* à la place de *EDF* que les échéances relatives sont grandes en comparaison des périodes (avec la garantie que $\varepsilon(DM) \geq 1/2$ dès lors que $D \geq T$ pour tout couple *(T,D)* de Υ) ; que les trafics sont homogènes ; ou que les échéances relatives deviennent petites face aux périodes (avec la garantie que $\varepsilon(DM)$ tend vers 1 lorsque la dispersion en période δT tend vers 1). Dans les autres cas, *EDF* reste l'algorithme le plus efficace sans que $\varepsilon(DM)$ ne puisse s'annuler.

Dans le cas non-préemptif, la relation d'équivalence en périodes et en échéances pour la faisabilité des trafics non-concrets τ ne tient plus. Il reste cependant possible de faire appel à l'algèbre linéaire pour mesurer l'efficacité des algorithmes à l'aide de conditions suffisantes pour majorer les inversions de priorités (cf. [Leboucher 1998]). Nous avons cherché à préciser si la supériorité théorique de *EDF*, significative en préemptif (cf. chapitre VI.1.3. p. 55), restait avérée lorsque l'on compare *NP-EDF* par rapport aux algorithmes à priorités fixes non-préemptifs non oisifs. Pour cela, sans établir de minorations sur l'efficacité des algorithmes non-préemptifs non-oisifs, nous avons utilisé N_{EDF} pour établir des bornes de faisabilité. Il apparaît que (cf. chapitre IX.1. p. 85) :

- la dispersion en échéances δD pénalise fortement les algorithmes non-préemptifs non-oisifs (que ce soit *NP-EDF* ou un algorithme à priorités fixes), notamment lorsque certaines échéances sont petites face aux périodes,
- la borne obtenue pour un algorithme non-préemptif non-oisif est proche de l'efficacité de sa version préemptive (que ce soit *EDF* ou un algorithme à priorités fixes) lorsque les couples *(T,D)* sont homogène ou lorsque les échéances sont grandes face aux périodes.

Contrairement au cas préemptif, et bien que *NP-EDF* reste Σ-optimal parmi les algorithmes non-préemptifs non-oisifs en présence de trafics non-concrets (cf. Théorème 23 p. 77), il semble donc qu'il faille relativiser l'interêt de cet algorithme face aux priorités fixes dans un tel référentiel d'ordonnancement.

Ces résultats d'efficacité en préemptif, comme en non-préemptif non-oisifs, ont été illustré par quelques applications numériques lors du chapitre X. p. 93.

XI.3. Coût des tests de faisabilité

Par définition le coût des tests de faisabilité (cf. Propriété 4 p. 12) dépend de l'énoncé choisi pour la faisabilité. Ainsi, chercher à établir la faisabilité de trafics concrets ω conduit rapidement à des problèmes NP-complets (citons le chapitre VII.1.1. p. 68, pour les algorithmes non-préemptifs oisifs où le chapitre IV.2.3.3. p. 31, pour *EDF* dans le cas $D_i \leq T_i$, $\forall i \in [1, n]$).

Pour les trafics non-concrets τ (cf. Hypothèse 1 p. 9) et les algorithmes non-oisifs considérés dans nos référentiels d'ordonnancement :

- l'état de l'art montre que les tests obtenus sont pseudo-polynomiaux (cf. chapitre VI.2.1. p. 61). Les trafics non-concrets τ "cassent" d'une certaine façon la complexité en rendant l'analyse de faisabilité plus pessimiste. Ils considèrent en effet tous les scénarii d'activation possibles mais se justifient cependant car il peut être intéressant de ne pas avoir à fixer les instants d'activation des tâches lors de la spécification du problème (cf. chapitre II.2.1.1. p. 8). De même, les algorithmes non-oisifs sont les plus implantés et ne font pas d'hypothèses sur le futur pour prendre les bonnes décisions d'ordonnancement.
- les majorants sur les coûts ne dépassent pas un ordre de $O(n^3)$ et ont été établis dans [Hermant et al. 1996] par notre co-auteur J. F. Hermant. Nous avons simplement montré que ces majorants sont similaires en préemptif et en non-préemptif non-oisif car le coût additionnel introduit par le calcul des inversions de priorités ne modifie pas l'ordre de grandeur (cf. chapitre IX.2. p. 89).
- pour les algorithmes à priorités fixes, il n'y a pas d'autre choix à notre connaissance que de passer par un calcul des pires temps de réponse des tâches (cf. chapitre V.2. p. 47). En conséquence, si la faisabilité seule du trafic est exigée, comme c'est le cas de la plupart des applications temps réel embarquées, le test de la faisabilité avec *EDF* est moins coûteux en pire cas ($O(n^2)$ contre $O(n^3)$, cf. chapitre VI.2.2. p. 62).
- si les pires temps de réponse sont exigés en plus de la faisabilité du trafic (comme c'est le cas par exemple des applications réparties mettant en oeuvre du contrôle de bout en bout [Tindell 1995]), le coût des tests est en $O(n^3)$. Le facteur constant est cependant plus faible pour les algorithmes à priorités fixes que pour *EDF*. Notons que si nous utilisions la procédure d'*Audsley* - optimale pour assigner les priorités fixes en présence de trafics généraux en préemptif et non-préemptif - il faudrait alors multiplier par $O(n^2)$ la complexité des résultats.

Conclusion

La théorie de l'ordonnancement temps réel, quoique largement étudiée depuis une vingtaine d'années, offre peu de critères de choix parmi les algorithmes pouvant résoudre un même problème. De nombreux résultats existent, qui cherchent plus à prendre en compte des contraintes spécifiques, qu'à se doter de critères de comparaison.

A l'inverse, nous nous en sommes tenus ici au "cas d'école" centralisé non-oisif, préemptif et non-préemptif, pour proposer une synthèse des résultats d'ordonnancement sur ces modèles génériques (pires temps de réponse des tâches, faisabilité des trafics, optimalité des algorithmes) ainsi que quelques éléments pour comparer les algorithmes (dominance, efficacité).

La synthèse proposée nous a conduit à utiliser certaines symétries sur les cas préemptifs/ non-préemptifs et priorités fixes/dynamiques pour identifier certains résultats d'ordonnancement manquants ou améliorations possibles.

Au-delà de cette synthèse, nous avons tenté de préciser l'intérêt théorique d'un algorithme à priorités dynamiques tel que *Earliest Deadline First* (noté *EDF*) face aux algorithmes à priorités fixes. L'optimalité s'avérant une propriété délicate, nous avons considéré le coût des tests de faisabilité, la dominance et surtout l'efficacité des algorithmes pour cette discussion.

Résumé

La partie A fixe le cahier des charges et les outils utilisés. Le chapitre 2 précise les hypothèses et propriétés utilisées pour notre modèle d'ordonnancement générique. Un trafic temps réel (noté τ) est constitué de tâches concurrentes pour l'accès à un serveur. Chaque tâche τ_i de τ est caractérisée par sa durée d'exécution et par un couple *(T,D)* où *T* est la période minimale d'activation de τ_i et *D* l'échéance de terminaison relative à chaque instant d'activation de τ_i (dans le cas général *T* est indépendant de *D*). Chaque trafic τ donne ainsi lieu à une infinité de scénarii d'activation (notés ω) en fixant les instants d'activation des tâches. Le chapitre 3 précise les outils utilisés. Au-delà des concepts classiques, il insiste sur les périodes occupées qui, grâce à la méthode du point fixe, permettront en les spécialisant par type de priorité (on introduit ainsi les *deadline busy periods* pour *EDF*) de calculer les instants d'exécution des activations de tâches. Des référentiels d'ordonnancement (notés Σ) sont utilisés pour fixer (1) le problème temps réel modélisé par un ensemble (noté Υ) de trafics non-concrets caractérisés par des couples *(T,D)*, (2) l'ensemble d'algorithmes (noté Π) candidats à résoudre ce problème, (3) les propriétés temps réel examinées sur ces algorithmes. En particulier la définition de la faisabilité choisie, à la base des autres propriétés, énonce qu'un trafic non-concret $\tau \in \Upsilon$ est faisable dans Σ si un même algorithme $Q \in \Pi$ peut ordonnancer tout scénario d'activation ω de τ. D'autres définitions possibles de la faisabilité sont discutées.

La partie B est consacrée au contexte préemptif. Le chapitre 4 propose un état de l'art pour les algorithmes à priorités fixes/dynamiques en utilisant pleinement les référentiels d'ordonnancement Σ pour énoncer sans ambiguïtés les résultats de faisabilité des trafics, de calculs de pires temps de réponse des tâches et d'optimalité des algorithmes. Cet état de l'art étant important, nous utilisons juste lors du chapitre 5 le concept de *deadline busy period* pour *EDF*, symétrique avec celui de *level-i busy period* utilisée avec les priorités fixes, pour améliorer les bornes sur l'intervalle d'étude par tâche τ_i de τ. Nous identifions ensuite quelques limitations sur l'optimalité des algorithmes à priorités fixes grâce à la propriété de dominance qui énonce qu'un algorithme en domine un autre si les trafics faisables dans Σ par le deuxième le sont aussi par le premier. La procédure d'*Audsley*, qui améliore une recherche exhaustive pour assigner les priorité fixes de façon optimale, reste coûteuse dans le cas général. *Deadline Monotonic* (noté *DM*) s'avère l'algorithme à priorités fixes dominant pour notre cahier des charges, quoique son optimalité reste limitée aussi bien sur Υ que sur Π. Le chapitre 6 évalue les algorithmes en mesurant l'efficacité (notée $\varepsilon(P)$ pour tout algorithme $P \in \Pi$) et en rappelant des majorants sur le coût des tests de faisabilité. L'efficacité[1], introduite dans [Hermant et al. 1996] par L. Leboucher, mesure en fait la proximité à l'optimalité de tout algorithme dans un référentiel Σ (ainsi ($\varepsilon(P) = 1$) => *P* est optimal dans Σ). $\varepsilon(EDF)$ étant égal à 1 pour tout Σ, nous spécialisons les minorations existantes, sur l'efficacité de tout algorithme à priorités fixes, pour l'algorithme à priorités fixes dominant *DM*. Il apparaît qu'il est d'autant plus pertinent d'utiliser *DM* à la place de *EDF* que les échéances relatives sont grandes face aux périodes (avec la garantie que $\varepsilon(DM) \geq 1/2$ dès lors que $D \geq T$ pour tout couple *(T,D)* de Υ) ; que les trafics sont homogènes (c.à.d. qu'il existe un seul couple *(T,D)* dans Υ) ; ou que les échéances relatives deviennent petites face aux périodes (avec la garantie que $\varepsilon(DM)$ tend alors vers 1 lorsque la dispersion en période $max(T_i)/min(T_i)$ tend aussi vers 1). Dans les autres cas, *EDF* reste l'algorithme le plus efficace sans que $\varepsilon(DM)$ ne puisse s'annuler.

1. qui généralise, à l'aide de l'algèbre linéaire, l'un des rares points de comparaison connus entre les performances des algorithmes, introduit (mais non exploité depuis) dans l'article de référence [Liu and Layland 1973].

La partie C est consacrée au contexte non-préemptif non-oisif. L'état de l'art proposé lors du chapitre 7 étant moins complet qu'en préemptif, nous mettons en avant lors du chapitre 8 les similitudes et les différences avec ce cas. Nous réutilisons ainsi les *deadline busy periods* avec la version non-préemptive non-oisive de *EDF* (notée *NP-EDF*) pour établir les pires temps de réponse des tâches τ_i de τ. Nous observons que les résultats d'optimalité sont plus limités qu'en préemptif, en particulier pour les priorités fixes, et que les pires temps de réponse ne sont pas nécessairement supérieurs, malgré les inversions de priorités possibles. *NP-EDF* reste optimal mais uniquement en présence de nos trafics τ et d'algorithmes non-préemptifs non oisifs. Nous montrons lors du chapitre 9 que cela ne se traduit plus, comme en préemptif, par une supériorité théorique significative en termes d'efficacité face aux algorithmes à priorités fixes. Il semble donc donc, contrairement au cas préemptif, qu'il faille relativiser l'intérêt de cet algorithme en termes d'optimalité comme d'efficacité dans le cas non-préemptif non-oisif. Nous constatons par contre que le coût des conditions de faisabilité reste proche du cas préemptif.

La partie D propose enfin, lors du chapitre 10, quelques applications numériques pour vérifier les résultats obtenus en termes d'efficacité et, lors du chapitre 11, une synthèse des résultats en référence aux propriétés posées dans notre cahier des charges.

Prolongements possibles

Les *deadline busy periods* introduites pour *EDF*, symétriques avec les *level-i busy periods* utilisées avec les algorithmes à priorités fixes, permettent d'identifier aisément les pires scénarii d'activation, les pires inversions de priorités en non-préemptif ou d'améliorer les bornes sur les intervalles d'étude. Une question ouverte subsiste : existe-t-il un calcul de ces bornes à faible coût, ou est-il possible de l'effectuer en-ligne comme avec les priorités fixes ?

L'optimalité s'avère une propriété délicate. Ainsi, de nombreux référentiels d'ordonnancement n'ont pas de résultats d'optimalité identifiés, en particulier avec les priorités fixes. Nous avons utilisé les propriétés de dominance et d'efficacité pour permettre la comparaison des algorithmes, mais n'avons pas obtenu de résultats nouveaux sur l'optimalité si ce n'est quelques impossibilités pratiques par des contres exemples. Les limites et les possibilités de cette propriété restent à explorer.

Les minorations examinées sur l'efficacité des algorithmes à priorités fixes restent pessimistes. Nous pensons qu'elles sont largement perfectibles pour permettre d'interpréter l'écart à l'optimalité dans un plus grand nombre de cas.

Les référentiels d'ordonnancement considérés couvrent des problèmes temps réel classiques pour lesquels nous avons tenté de systématiser et de compléter les résultats existants. D'une part ce concept de référentiel d'ordonnancement, introduit dans [Hermant et al. 1996] par L. Leboucher, n'est limité ni sur les trafics, ni sur les algorithmes, ni sur les propriétés que l'on veut examiner. D'autre part, il permet de poser de façon rigoureuse ces différents points. Notre objectif est donc de continuer à utiliser cet outil pour explorer des problèmes d'ordonnancement temps réel variés. A titre d'exemple, lors de l'annexe B, p. 121, nous amorçons la discussion pour prendre en compte la gigue d'activation et les facteurs de blocage.

Glossaire

activation	cf. chapitre II.2.1.1. p. 8.
Audsley	cf. chapitre IV.3.4.3. p. 40.
C_i	Durée d'exécution de τ_i (cf. Définition 1 p. 8).
cahier des charges	cf. chapitre II.2. p. 8.
charge	cf. Définition 9 p. 15.
charge cumulée	cf. Définition 14 p. 15.
concret	cf. chapitre II.2.1.1. p. 8.
couple *(T,D)*	un couple période-échéance (cf. Définition 27 p. 22 et Définition 36 p. 52).
D_i	Echéance relative de τ_i (cf. Définition 1 p. 8).
$\{D_i\}$ » $\{T_i\}$	Lorsque toutes les échéances relatives dans τ sont supérieures aux périodes.
$\{D_i\}$ « $\{T_i\}$	Lorsque toutes les échéances relatives dans τ sont inférieures aux périodes.
deadline busy period	cf. Définition 19 p. 19.
demande processeur	cf. Définition 15 p. 16.
dispersion	cf. δD, δT.
dominance	cf. Définition 35 p. 48.
dispatcheur	cf. Définition 8 p. 13.
DM	*Deadline Monotonic* (cf. chapitre IV.3.3.2. p. 37).
EDF	*Earliest Deadline First* (cf. chapitre IV.2.1. p. 28).
efficacité	cf. Définition 7 p. 11, Définition 38 p. 53.
ETPN	Ensemble de Trafics Périodiques Non-concrets (Définition 27 p. 22)
faisabilité	cf. Définition 4 p. 10, Définition 23 p. 21 et chapitre III.3.2. p. 22.
homogène	$\forall i$, $D_i=D$, $T_i=T$ (cf. chapitre VI.1.3.2.b p. 57).
général	cf. chapitre II.2.1.1. p. 8.
$h(t)$	La demande processeur à l'instant *t* (cf. Définition 15 p. 16).
HPF	*Highest Priority First* (cf. Définition 8 p. 13).
instant synchrone	cf. Définition 11 p. 15.
L	Taille de la période occupée synchrone du processeur (cf. Définition 17 p. 17).
L_i	Taille maximale de la période occupée de priorité pour la tâche τ_i (cf. chapitre III.2.2. p. 19).
$L_i(a)$	Taille d'une *deadline(a+D_i) busy period* (cf. chapitre III.2.2. p. 19).
level-i busy period	cf. Définition 18 p. 19.
n	Nombre de tâche d'un trafic τ.
nécessaire	cf. chapitre II.3.2. p. 13.
non-concret	cf. chapitre II.2.1.1. p. 8.
non-oisif	cf. Hypothèse 2 p. 10.
non-préemptif	cf. Hypothèse 2 p. 10.
N_{EDF}	Norme *EDF* (cf. Théorème 17 p. 54).

NP-Q	La version Non-Préemptive non-oisive d'un algorithme *Q* (cf. chapitre II.3.1. p. 12).
N_U	Norme charge (cf. Théorème 17 p. 54).
N'_U	cf. Equation (20) p. 56.
optimalité	cf. Définition 6 p. 11, Définition 24 p. 21 et chapitre III.3.2. p. 22.
oisif	cf. Hypothèse 2 p. 10.
période de base	cf. Définition 13 p. 15.
période occupée	du processeur (cf. chapitre III.2.1. p. 17) ; de priorités maximales (cf. chapitre III.2.2. p. 19).
périodique	cf. Définition 1 p. 8.
préemptif	cf. Hypothèse 2 p. 10.
priorité	cf. Définition 8 p. 13.
pseudo-polynomial	cf. Propriété 4 p. 12.
r_i	Pire temps de réponse de τ_i (cf. Définition 5 p. 10).
s_i	L'instant de première activation de ω_i (cf. Définition 1 p. 8).
référentiel d'ordonnancement	cf. Définition 21 p. 20.
RM	*Rate Monotonic* (cf. chapitre IV.3.2.2. p. 36).
scénario critique	cf. Définition 10 p. 15.
scénario synchrone	cf. Définition 11 p. 15.
sporadique	cf. chapitre II.2.1.1. p. 8.
suffisant	cf. chapitre II.3.2. p. 13.
T_i	Période de τ_i (cf. Définition 1 p. 8).
tâche	cf. chapitre II.2.1.1. p. 8.
test	cf. Propriété 4 p. 12 et chapitre II.3.2. p. 13.
(T,D)	un couple période-échéance (cf. Définition 27 p. 22 et Définition 36 p. 52).
trafic	cf. chapitre II.2.1.1. p. 8.
TPN	Trafic Périodique Non-concret (cf. Définition 27 p. 22).
TRDF	cf. chapitre II.1. p. 7.
U	La charge d'un trafic τ (cf. Définition 9 p. 15).
valide	cf. Définition 3 p. 10.
$W(t)$	La charge cumulée à l'instant *t* (cf. Définition 14 p. 15).
$w_{i,q}$	Taille d'une *level-i busy period* (cf. chapitre III.2.2. p. 19).
$\varepsilon(Q)$	Efficacité de l'algorithme *Q* (cf. Définition 38 p. 53).
δD	Dispersion en échéances : $max(D_i)/min(D_i)$.
δT	Dispersion en périodes : $max(T_i)/min(T_i)$.
Π	Classe d'algorithmes dans le référentiel Σ (cf. Définition 21 p. 20).
Σ	Référentiel d'ordonnancement (cf. Définition 21 p. 20).
τ	Trafic non-concret (cf. Définition 2 p. 9).
τ_i	Tache non-concrète (cf. Définition 1 p. 8).
Υ	Classe de trafics dans le référentiel Σ (cf. Définition 21 p. 20).
ω	Trafic concret (cf. Définition 2 p. 9).
ω_i	Tache concrète (cf. Définition 1 p. 8).

Annexes

Annexe A. Calculs de faisabilité et d'efficacité en préemptif

Cette annexe illustre les calculs de faisabilité, de pires temps de réponse et de normes valides sur quelques trafics non-concrets (cf. Définition 2, page 9). Elle reprend ensuite les couples *(T,D)* de ces trafics et illustre les calculs d'efficacité dans les référentiels d'ordonnancement résultants.

Annexe A. 1. Mesures

Soit les trafics non-concrets τ suivants (où $\tau_i=(C_i, D_i, T_i)$),

- $\tau(1)$ = ((3, 20, 12) ; (4, 20, 12) ; (1, 20, 12) ; (1, 20, 12) ; (1, 20, 12) ; (2, 20, 12)).
- $\tau(2)$ = ((12, 20, 12)).
- $\tau(3)$ = ((15, 30, 40) ; (15, 30, 40)).
- $\tau(4)$ = ((2, 20, 12) ; (2, 20, 12) ; (2, 20, 12) ; (15, 30, 40)).
- $\tau(5)$ = ((2, 5, 7) ; (3, 7, 11) ; (5, 10, 13)).
- $\tau(6)$ = ((1, 5, 10) ; (1, 13, 18) ; (5, 20, 45) ; (9, 40, 30) ; (7, 45, 32) ; (11, 80, 150) ; (4, 180, 50)).
- $\tau(7)$ = ((2227, 5000, 200000); (1423, 12000, 25000); (420, 14199, 40000);
  (496, 19199, 20000); (552, 19199, 160000); (3096, 50000, 50000);
  (7880, 59000, 59000); (3220, 87199, 800000); (3220, 98399, 100000);
  (1996, 100000, 50000); (520, 100000, 200000); (1990, 193499, 1000000);
  (1120, 197598, 200000); (954, 197598, 2000000); (1124, 198545, 200000);
  (3345, 200000, 200000)).

La table suivante donne pour chacun de ces trafics :

- r_{DM}, les pires temps de réponse obtenus par *DM* (cf. Théorème 14 p. 50). (Diagnostic sur le respect des échéances : succès ou échec)
- r_{EDF}, les pires temps de réponse obtenus par *EDF* (cf. Théorème 13 p. 47). (Diagnostic sur le respect des échéances : succès ou échec).
- la norme valide $N_{EDF}(\tau)$ (du Théorème 17 p. 54), $N_U(\tau)$ (du même théorème) et $N'_U(\tau)$ (de l'Equation (20), page 56)

Table 11 : Faisabilité des trafics, pires temps de réponse et normes valides

τ	r_{DM}	r_{EDF}	$N_{EDF}(\tau)$, $N_U(\tau)$, $N'_U(\tau)$
(1)	r_1=3, r_2=7, r_3=8, r_4=9, r_5=10,r_6=12 (succès)	r_1=12, r_2=12, r_3=12, r_4=12,r_5=12,r_6=12 (succès)	1, 1, 1
(2)	r_1=12 (succès)	r_1=12 (succès)	1, 1, 1
(3)	r_1=30 (succès)	r_1=30 (succès)	1, 0.75, 1
(4)	r_1=6, r_2=33 (échec)	r_1=15, r_2=25 (succès)	0.875, 0.875, 0.875
(5)	r_1=2, r_2=5, r_3=17 (échec)	r_1=5, r_2=7, r_3=10 (succès	1, 0.943, 0.943
(6)	r_1=1, r_2=2, r_3=7, r_4=17, r_5=26, r_6=83, r_7=87 (échec)	r_1=1, r_2=2, r_3=7, r_4=24, r_5=29, r_6=64, r_7=87 (succès)	0.939, 0.939, 0.939
(7)	r_1=2227, r_2=3650, r_3=4070, r_4=4566, r_5=5118, r_6=8214, r_7=16094, r_8=19314, r_9=23030, r_{10}=26449 r_{11}=26969, r_{12}=28959, r_{13}=30079, r_{14}=31033, r_{15}=32157, r_{16}=35502 (succès)	r_1=2227, r_2=3650, r_3=4070, r_4=4566, r_5=5118, r_6=8214, r_7=16094, r_8=19314 r_9=25368, r_{10}=26969, r_{11}=26969, r_{12}=29001, r_{13}=33100, r_{14}=33100 r_{15}=34047r_{16}=35502 (succès)	0.445, 0.411, 0.411

Soit $\Sigma = (\Upsilon, \Pi)$, le référentiel d'ordonnancement où $\Pi = \{EDF, DM...\}$ et Υ est caractérisé par les couples[1] (*T*,*D*) d'un des trafics précédents. La table suivante donne :

- $\varepsilon(EDF)$, l'efficacité de *EDF* dans Σ toujours égale à 1 (cf. Théorème 17 p. 54).
- $\varepsilon(DM) = \alpha_{N_{EDF}}(DM)$, le calcul exact de la N_{EDF}-efficacité de *DM* dans Σ (cf. [Hermant et al. 1996] pour la procédure de calcul).
  $\alpha_{N_U}(DM)$, le calcul exact de la N_U-efficacité de *DM* dans Σ.
  $\alpha_{N'_U}(DM)$, le calcul exact de la N'_U-efficacité de *DM* dans Σ.
- Minor1, une minoration de $\varepsilon(X)$, l'efficacité de tout algorithme d'ordonnancement à priorités fixes dans Σ (cf. Théorème 18 p. 56).
  Minor2, une minoration de $\varepsilon(DM)$, l'efficacité de *DM* dans Σ (cf. Théorème 19 p. 59).

1. Rappelons que les couples *(T,D)* d'un référentiel d'ordonnancement Σ ne limitent pas le nombre de tâches mais imposent à celles-ci les valeurs des périodes et des échéances relatives (cf. Définition 27, page 22).

Table 12 : Efficacité de *EDF* et *DM* dans Σ

Σ	$\varepsilon(EDF)$	$\varepsilon(DM)$, $\alpha_{N_U}(DM)$, $\alpha_{N'_U}(DM)$	Minor1, Minor2
(1)	1	1, 1, 1	0.625, 0.625
(2)	1	1, 1, 1	0.625, 0.625
(3)	1	1, 0.75, 1	0.571, 0.571
(4)	1	0.817, 0.775, 0.775	0.333, 0.428
(5)	1	0.917, 0.727, 0.727	0.278, 0.3888
(6)	1	0.773, 0.467, 0.467	0.032, 0.308
(7)	1		0.002, 0.09

Annexe A. 2. Discussion

Trafics τ(1), τ(2) et référentiel d'ordonnancement correspondant

Le trafic τ(2) est équivalent en périodes et en échéances à τ(1) (cf. Definition 36, page 52). Les résultats de faisabilité et de normes valides sont donc identiques. Les seules différences portent sur le pire temps de réponse de l'unique tâche de τ(2) égal au pire temps de réponse maximal dans τ(1), et sur le coût des calculs plus faible pour τ(2). Cette propriété simplificatrice sera donc toujours utilisée dans la suite. τ(2) est homogène, c.à.d. contient un seul couple (*T*,*D*) pouvant être vu comme un flux audio (si la granularité est de 1ms alors *D*=20ms ce qui est largement acceptable pour de la téléconférence [Jeffay et al. 1994]).

Comme attendu, *EDF* n'est pas plus efficace que *DM* dans le cas homogène puisque $\varepsilon(DM)=\varepsilon(EDF)=1$ (cf. chapitre VI.1.3.2.b, page 57). Comme de plus, *D*=20 est supérieur à *T*=12, nous avons $N_{EDF}(\tau)=N_U(\tau)=N'_U(\tau)$ (cf. Théorème 2 p. 30, pour N_U et N_{EDF} et Equation (20), page 56, pour N_U et N'_U). La faisabilité est alors simplement vérifiée par $\sum C_i = C \leq T$ (c.à.d. $N_U(\tau) \leq 1$).

Trafic τ(3) et référentiel d'ordonnancement correspondant

τ(3) peut être vu comme un flux vidéo que nous simplifions grâce à la propriété d'équivalence en périodes et en échéances par $\tau(3)=\{(30, 30, 40)\}$. Comme précédemment τ(3) est homogène, *EDF* n'est donc pas plus efficace que *DM*. Par contre, *D*=30 étant plus petit que *T*=40, l'analyse de faisabilité est maintenant simplifiée par $\sum C_i = C \leq D$ (cf. chapitre VI.1.3.2.b, page 57) et $\alpha_{N_U}(DM) \leq \varepsilon(DM) = 1$ puisque $\forall(\tau \in \Upsilon)$, $N_U(\tau) \leq N_{EDF}(\tau)$.

En d'autres termes, le fait que N_U ne soit pas un critère le plus fin (et donc la meilleure mesure de l'efficacité de *DM*) quand les échéances relatives sont inférieures aux périodes, est clairement visible sur ce simple exemple ($\varepsilon(DM)=1$ alors que $\alpha_{N_U}(DM)=0.75$). Par contre, N'_U, la norme valide introduite pour le théorème d'efficacité (cf. chapitre VI.1.3.2.a, page 56), conduit ici à une mesure facile et précise de l'efficacité de *DM* ($\varepsilon(DM)=\alpha_{N'_U}(DM)=1$).

Trafic τ(4) et référentiel d'ordonnancement correspondant

τ(4) correspond à la combinaison linéaire de (τ(2)+τ(3))/2, c.à.d. peut être vu comme un mélange de flux audio et vidéo. Grâce à la propriété d'équivalence en périodes et en échéances, nous simplifions ce trafic par $\tau(4)=\{(6, 20, 12);(15, 30, 40)\}$. τ(2) et τ(3) étant faisables par *EDF*, τ(4) doit être encore faisable par *EDF* (cf. chapitre VI.1.2., page 52), ce qui est vérifié par le calcul des pires temps de réponse avec *EDF*.

Par contre $\tau(4)$ étant un trafic hétérogène, *DM* doit être moins efficace que *EDF* (cf. chapitre VI.1.3.2.b, page 57), ce qui est aussi vérifié par $\varepsilon(EDF) = 1 \geq \varepsilon(DM) = 0,817$. En particulier $\tau(4)$ n'est pas faisable par *DM* puisque r_2=33>30. Notons que si nous diminuons la durée d'exécution de la deuxième tâche de façon à obtenir $\tau'(4)$={(6, 20, 12);(12, 30, 40)}, le trafic devient alors faisable par *DM*. En utilisant N_{EDF} (le critère d'optimalité), ceci était prévisible puisque $N_{EDF}(\tau'(4)) = 0,8$ inférieur à $\varepsilon(DM) < 0,817$. Par contre en utilisant N_U (qui n'est pas un critère d'optimalité), ceci n'était pas prévisible puisque $\alpha_{N_U}(DM) = 0,775$. La même remarque tient pour N'_U puisque le deuxième couple (*T*, *D*) présente une échéance relative supérieure à la période (c.à.d. $N'_U = N_U$, cf. chapitre VI.1.3.2.a, page 56).

Trafic $\tau(5)$, $\tau(6)$ et référentiels d'ordonnancement correspondants

$\tau(5)$ et $\tau(6)$ modélisent de petites applications temps réel embarquées. Dans les deux cas la charge est proche de 1 et ces trafics sont faisables par *EDF* car $N_{EDF}(\tau) \leq 1$. Par contre $N_{EDF}(\tau)$ (respectivement $N_U(\tau)$) étant supérieur à $\varepsilon(DM)$ (respectivement à $\alpha_{N_U}(DM)$), il est impossible de conclure à la faisabilité par *DM* par ces mesures. Le calcul des pires temps de réponse nous apprend d'ailleurs que ces deux trafics ne sont pas faisables par *DM*.

Une remarque plus générale (prévue lors du chapitre VI.1.3.2.b, page 57 et vérifiée par la minoration sur $\varepsilon(P)$) est que les couples *(T,D)* de $\tau(6)$ étant plus hétérogène que ceux de $\tau(5)$ (cf. la dispersion en échéance et en période), l'efficacité de *DM* doit être plus faible pour $\tau(6)$. Ceci est vérifié par le calcul exact de $\varepsilon(DM)$ et $\alpha_{N'_U}(DM)$. Comme attendu, les minorations proposées sur l'efficacité des algorithmes à priorités fixes sont pessimistes. Notons toutefois que Minor1 porte sur tout algorithme à priorités fixes, sa valeur est donc inférieure à Minor2 qui minore uniquement l'efficacité de *DM*. Rappelons que nous considérons avant tout les théorèmes d'efficacité comme des procédures rapides pour évaluer le comportement asymptotique des algorithmes d'ordonnancement et que des raffinements sont toujours possibles sur ces minorants.

Trafic $\tau(7)$ et référentiel d'ordonnancement correspondant

$\tau(7)$ enfin est un exemple repris de [Clark and Tindell 1994] et [Spuri 1996b] qui décrivent un hypothétique système de contrôle aérien. Pour un tel trafic, la procédure proposée dans [Hermant et al. 1996] est trop coûteuse pour calculer exactement l'efficacité de *DM* (des améliorations seront proposées dans [Hermant 1998], [Leboucher 1998]). Par contre la (forte) minoration sur $\varepsilon(P)$ obtenue par les théorèmes d'efficacité (cf. chapitre VI.1.3.2.a, page 56) semble montrer que dans ce cas *DM* est largement moins efficace que par *EDF* en terme d'efficacité.

Annexe B. Variantes sur le cahier des charges

Les hypothèses simples énoncées dans notre cahier des charges (cf. chapitre II.2.1., page 8) nous ont conduit à rappeler et développer quelques résultats génériques en ordonnancement temps réel centralisé préemptif puis non-préemptif non-oisif. De nombreuses variantes du problème posé sont possibles. Nous avons déjà vu lors du chapitre III.3.2., page 22, qu'il était possible de jouer sur la définition de la faisabilité. La conclusion étant que le choix de la "bonne" définition ne dépendait pas tant du "fournisseur" de solutions algorithmiques que du problème posé par le client.

De la même façon, il est possible de jouer sur le référentiel d'ordonnancement Σ, aussi bien sur Υ que sur Π. A titre d'exemple nous allons examiner ici :

- la gigue d'activation qui modifie les trafics dans Υ (cf. annexe B. 1, p. 121).
- les facteurs de blocage qui modifie l'algorithmique dans Π (cf. annexe B. 2, p. 123).

L'idée n'est pas de faire le tour de ces questions mais simplement de montrer que la prise en compte de contraintes applicatives différente peut s'appuyer sur les résultats génériques précédents. Les implications en termes de faisabilité, d'optimalité, d'efficacité et de coûts sont brièvement discutées afin de proposer quelques prolongements à ce travail.

Annexe B. 1. Gigue d'activation

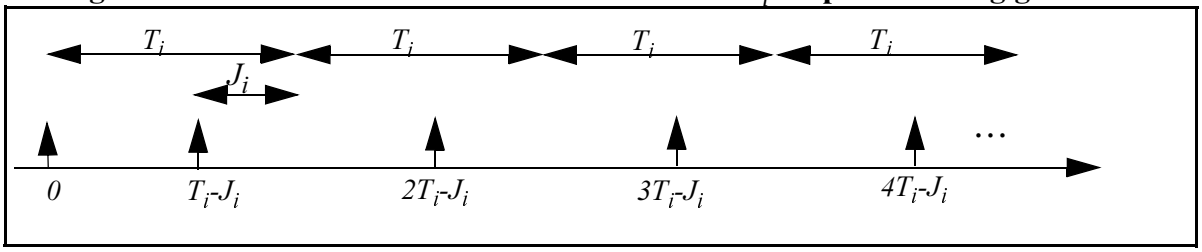
Annexe B.1.1. Le problème

Les résultats précédents considèrent des trafics non-concrets τ (cf. Hypothèse 1, page 9) qui n'imposent pas d'hypothèses irréalistes, telles que la connaissance des instants d'activation ou de périodicité stricte. Tout au plus, le client s'engage sur une densité maximale d'activations par tâche. Ils n'imposent pas non plus de relations particulières entre les paramètres des tâches. Les résultats obtenus en présence de telles relations (par exemple $\forall i \in [1, n],\ D_i = T_i$) sont donc vus comme des cas particuliers.

Pour des raisons de *tick scheduling* (les décisions d'ordonnancement sont généralement faites sur des tops d'horloge dans un système d'exploitation) ou de contexte réparti, il peut être nécessaire d'enrichir ce modèle de trafic en introduisant une gigue d'activation. Ainsi, l'approche holistique introduite par [Tindell 1995] en présence de priorités fixes, puis étendue par [Spuri 1996b] pour *EDF*, énonce la faisabilité d'un système réparti, de type chaîne de traitements. Les pires temps de réponse engendrés par chaque maillon de la chaîne sont utilisés sous forme de gigue d'activation lors du maillon suivant.

Pour modéliser cette gigue (J_i pour la tâche τ_i), nous considérerons ici que si la *1ère* activation de τ_i arrive en 0 alors la *(k+1)ième* activation de τ_i arrive au minimum à l'instant $t_{k+1} = kT_i - J_i$. $[T_i - J_i]$ représente alors l'intervalle minimal entre deux activations de la même tâche (au lieu de l'intervalle $[T_i]$ en absence de gigue). Notons que cette inter-arrivée minimale ne peut être cumulée (cf. Figure 26).

Figure 26 : Maximisation de la densité de la tâche τ_i en présence de gigue



Annexe B.1.2. Discussion

La densité d'activation et d'échéance absolue de la tâche τ_i est donc maximisée en appliquant cette gigue entre la première et la deuxième activation de τ_i et en réactivant la tâche périodiquement ensuite. Avec ce décalage le plus à gauche possible, il y a alors à l'instant *t* $\lceil (t + J_i)/T_i \rceil$ activations, et $max(0, 1 + \lfloor (t + J_i - D_i)/T_i \rfloor)$ échéances absolues, de la tâche τ_i (à comparer à $\lceil t/T_i \rceil$ et $max(0, 1 + \lfloor (t - D_i)/T_i \rfloor)$ en l'absence de gigue). Il suit que *L,* la longueur maximale de la période occupée du processeur, est obtenue par l'équation récursive $L = \sum_i \lceil (L + J_i)/T_i \rceil C_j$ au lieu de $L = \sum_i \lceil L/T_i \rceil C_j$ sans gigue (cf. Equation (1), page 17).

L'impact sur les résultats précédents est limité. Les référentiels d'ordonnancement sont simplement modifiés par la prise en compte de trafics non-concrets τ avec gigues dans Υ. Les scénarii d'activation pires cas et les conditions de faisabilité se déduisent trivialement des résultats précédents en tenant cependant compte du décalage le plus à gauche possible des activations et des échéances absolues. Les preuves d'optimalité sont les mêmes, les résultats d'efficacité aussi en s'appuyant cependant sur les triplets *(T,D,J)* et non pas simplement sur les couples *(T,D)* (cf. Définition 27, page 22). Ainsi, en préemptif, nous avons la norme suivante (à comparer avec celle du Theoreme 17, page 54) :

$$N_{EDF}(\tau) = sup_{t \in \mathbb{R}^+} \left\{ \frac{1}{t} \sum_{(T,D,J) \in T \times D} max\left\{ 0, 1 + \left\lfloor \frac{t + J - D}{T} \right\rfloor \right\} \middle| C_{T,D,J} \right\}.$$

Enfin le coût des tests n'est pas modifié puisque l'on a toujours à considérer des prédicats en *O(n)* et les mêmes majorations en nombre d'itération ou de points de discontinuité.

Pour l'énoncé des tests, dans le cas préemptif à priorités fixes, [Burns et al. 1994] déduisent des résultats de la partie B un test nécéssaire et suffisant pour trafics généraux avec gigue. [Spuri 1996] adapte ce résultat pour *EDF*.

A titre d'exemple, dans le cas non-préemptif, pour la faisabilité par *NP-EDF* nous obtenons de même trivialement.

Lemme 23 - *Soit un trafic non-concret τ avec gigue ordonnancé par NP-EDF. Si une échéance absolue est ratée pour un certain scénario d'activation, alors une deadline-t busy period, résultant du scénario d'activation suivant, conduit à rater une échéance en t. Toutes les tâches sont activées périodiquement et appliquent leurs gigues entre la première et la deuxième activation. La tâche τ_j, vérifiant $B(t) = max_{D_j - J_j > t}\{C_j - 1\}$, est activée à l'instant -1 et toutes les autres tâches sont synchrones en 0 (par convention, toutes les tâches sont synchrones en 0 si $B(t) = 0$)*

Preuve. cf. Lemme 17, page 73, en tenant compte de la gigue. ❐

Théorème 27 - *Soit un référentiel d'ordonnancement $\Sigma=(\Upsilon,\Pi)$ où Υ contient des trafics généraux avec gigue et Π contient NP-EDF. $\forall(\tau \in \Upsilon)$, τ est NP-EDF-faisable si et seulement si :*

$$\forall t \le L, \quad \sum_{D_i - J_i \le t} (1 + \lfloor (t + J_i - D_i)/T_i \rfloor) C_i + max_{D_j - J_j > t}\{C_j - 1\} \le t,$$

avec la convention que $max_{D_i - J_i > t}\{C_i - 1\} = 0$ si $\nexists i: D_i - J_i > t$

et $L = \sum_{i=1}^{n} \lceil (L + J_i)/T_i \rceil C_j$ (la première racine de cette équation)

Preuve. Immédiat du Lemme 23 et du Théorème 23 p. 77. ❐

Annexe B. 2. Facteurs de blocage

Annexe B.2.1. Le problème

Les modèles examinés lors des parties B & C considèrent des tâches non-concrètes qui ne partagent pas de ressources, exepté le serveur (cf. Hypothèse 1, page 9). Dans le cas préemptif et sans ressources partagées (cf. Partie B), nous avons vu que l'ordonnancement suit exactement l'assignation de priorités choisi (*RM*, *EDF*...). En présence de ressources partagées (par exemple un fichier), des méchanismes spécifiques peuvent être intégrés dans les algorithmes d'ordonnancement afin d'assurer la cohérence de ces resources. Ainsi une activation de tâche en file d'attente pourra se voir refuser l'accès à la préemption par le *dispatcheur* (cf. Définition 8, page 13), si une activation de tâche de plus basse priorité, mais en cours d'exécution, est entrée en section critique sur une ressource.

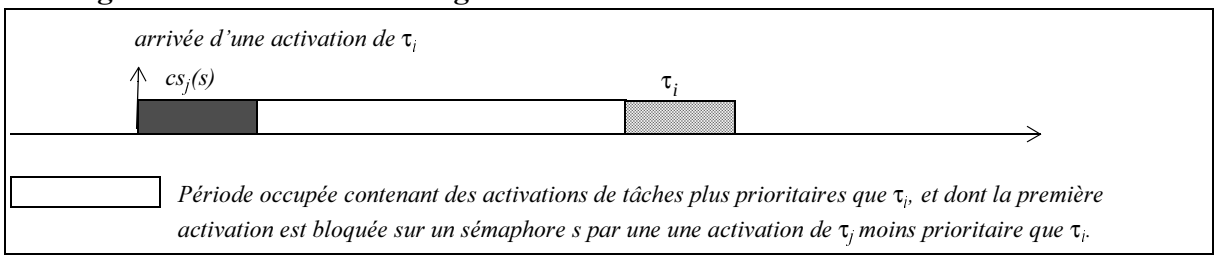
La durée maximale de ces blocages (des inversions de priorités comme en non-préemptif) peut être bornée si les ressources partagées sont accédées par un protocole de verrouillage comme le *priority ceiling* (cf. [Lehoczky et al. 1990b] avec les priorités fixes et [Chen and Lin1990] avec *EDF*) ou le *stack resource policy* (cf. [Baker 1991] avec des priorités fixes ou dynamiques). Sans détailler ces algorithmes nous discutons brièvement ici de leurs implications en termes de facteur de blocage sur les résultats précédents.

L'avantage de ces protocoles est d'interdire tout *deadlock* et de n'autoriser qu'un blocage par activation de tâche. Il est ainsi possible de calculer la durée maximale où une activation de tâche peut être retardée, soit directement par une activation de tâche de priorité inférieure entrée en section critique, soit indirectement par une même activation ayant retardée préalablement une ou des activations de tâches de priorités supérieures. A titre d'exemple, [Baker 1991] formalise ceci en introduisant π_i, le *preemption level* de τ_i (une activation de τ_i n'est autorisée à préempter une activation τ_j que si $\pi_i > \pi_j$). Connaissant $cs_j(s)$, la durée du blocage introduit par une activation de tâche τ_j lorsqu'elle verrouille un sémaphore *s*, Baker en déduit alors B_i, le blocage maximum de τ_i par tout *s*, comme suit (cf. Figure 27) :

$$B_i = max_{j,s}\{cs_j(s) \mid (\lceil s \rceil \geq \pi_i \wedge \pi_j < \pi_i)\} \text{ où } \lceil s \rceil = max\{\pi_k \mid \tau_k \text{ may lock } s\}. \quad (43)$$

En d'autres termes, B_i est égal à la durée de blocage maximale sur *s* provoquée par une tâche τ_j de *preemption level* inférieur à τ_i et tel que l'une des tâches de *preemption level* supérieur ou égal à τ_i puisse être bloquée par un sémaphore *s* (B_i s'annule sinon).

Figure 27 : Facteur de blocage



Annexe B.2.2. Discussion

Dans le cas non-préemptif avec partage de ressources, des inversions de priorités peuvent se produire (cf. Partie C) mais il ne peut y avoir par définition d'activations de tâches concurrentes sur une même ressource. Les résultats s'appliquent donc sans modifications. Dans le cas préemptif, par contre, les conditions de faisabilité doivent prendre en compte le terme additionel B_i. Il est alors intéressant de faire le rapprochement avec l'adaptation des résultats préemptifs de la Partie B vers ceux non-préemptifs non-oisifs de la Partie C.

Tout d'abord, contrairement à la gigue, la prise en compte de facteurs de blocage ne modifie pas la taille des périodes occupées du processeur dont les calculs ne font appel qu'au scénario d'activation. Ainsi L, la durée maximale de la période occupée du processeur servant à borner l'intervalle d'étude, est toujours obtenue par le scénario synchrone (cf. Lemme 2, page 17). Par contre, l'ordonnancement des périodes occupées de priorités (*level-i busy period* ou *deadline-d busy period*, cf. chapitre III.2.2., page 19) est perturbé par le terme B_i de l'Equation (43), page 123. Les conditions de faisabilité doivent donc en tenir compte. Notons que l'analogie avec le passage du préemptif au non-préemptif s'arrête là. D'une part, la taille des facteurs de blocage est calculée différemment. D'autre part, les périodes occupées de priorités maximales ne se terminent pas avant le début d'exécution de l'activation courante, comme c'est le cas en non-préemptif, mais sur la fin d'exécution de celle-ci, comme c'est le cas en préemptif.

En présence de priorités fixes, nous avons $\pi_i > \pi_j$ équivalent à $j \in lp(i)$ (où *lp(i)* dénote les tâches ayant une priorité inférieures à τ_i, par exemple $i < j$ si les tâches ayant les indices les plus faibles sont les plus prioritaires). Pour le calcul des pires temps de réponse en présence de priorités fixes et de ressources partagées, [Burns et al. 1994] montrent alors qu'il suffit simplement d'adapter le Théorème 14 p. 50 (qui identifie les plus longues *level-i busy periods* en préemptif, cf. partie B), en y ajoutant le terme B_i dans le calcul de $w_{i,q}$, comme suit:

$$w_{i,q} = (q+1)C_i + \sum_{j \in hp(i)} \lceil w_{i,q} / T_j \rceil C_j + B_i .$$

Avec *EDF*, il est intéressant de remarquer que l'attribution des *preemption level* se fait hors-ligne bien que les priorités (des échéances absolues) soient dynamiques. En effet $\pi_i > \pi_j \Leftrightarrow D_i < D_j$ signifie qu'une tâche ne peut jamais en préempter une autre si son échéance relative est supérieure. L'adaptation des résultats préemptifs conduit alors à déduire de chaque échéance absolue le terme B_i adéquat. [Spuri 1996] montre que si les indices des tâches sont attribués par échéances relatives croissantes, alors une condition de faisabilité suffisante en présence de trafics généraux et de ressources partagées se déduit ainsi du Theoreme 12, page 46 :

$$\forall t \le L,\ \sum_{D_i \le t} (1 + \lfloor (t - D_i)/T_i \rfloor) C_i + B_{k(t)} \le t \text{ où } k(t) = max\{k | (D_k \le t)\} .$$

Remarquons que contrairement aux priorités fixes, le blocage doit s'annuler lorsque $t > max\{D_i\}$ puisque, par définition, la demande processeur inclut alors tout blocage provoqué par une activation de tâche de priorité inférieure. Ceci est une propriété que nous avons déjà rencontrée en non-préemptif avec *NP-EDF* et que nous avons formalisée grâce au concept de *deadline-d busy period*. De la même façon [Spuri 1996] adapte le calcul des pires temps de réponse avec *EDF* en présence de ressource partagées.

Comme nous pouvons le constater, le coût des tests de la partie B n'est que très légèrement modifié puisque le calcul des B_i est effectué une fois pour toute, puis introduit sous la forme d'un terme additionnel dans les prédicats à tester. Par contre, contrairement à la gigue, l'impact est important sur l'optimalité et l'efficacité puisque l'algorithmique est modifiée. Ce problème est une question ouverte que nous nous contentons de formaliser, sous deux niveaux de difficulté, grâce au concept de référentiel d'ordonnancement :

- *Soit un référentiel d'ordonnancement* $\Sigma=(\Upsilon,\Pi)$ *où* Υ *contient des trafics généraux et* Π *contient tout algorithme d'ordonnancement préemptif P et* ***un*** *algorithme de partage de ressources R.*
- *Soit un référentiel d'ordonnancement* $\Sigma=(\Upsilon,\Pi)$ *où* Υ *contient des trafics généraux et* Π *contient tout algorithme d'ordonnancement préemptif P et* ***tout*** *algorithme de partage de ressources R.*

Dans ces deux cas, il faut trouver le couple *P/R* optimal et mesurer l'efficacité des autres. Le premier référentiel cependant ne nous éloigne pas trop des résultats précédents puisque *R* impose une même contrainte pour tout *P*, c'est donc l'algorithme d'ordonnancement *P* qui fait la différence. Par contre le deuxième référentiel pose le problème, non résolu en notre connaissance, de l'optimalité parmi les algorithmes de partage de ressources.

Bibliographie

Eléments de bibliographie

[Gennaro et Rivierre1993]
Gennaro, B. Rivierre, N. Sept. 1993. Schedulability analysis over broadcast networks. IEEE. FTDCS, Lisbon. 332-338.

[George et al. 1996]
George, L. Rivierre, N. Spuri, M. 1996. Preemptive and non-preemptive real-time uniprocessor scheduling. RR-2966, INRIA, France.

[Hermant et al. 1995]
Hermant, J.F. Le Lann, G. Rivierre, Oct. 1995. A general approach to real-time scheduling over broadcast channels. IEEE/INRIA, ETFA'95. 191-204.

[Hermant et al. 1996]
Hermant, J.F. Leboucher, L. Rivierre, N. 1996. Real-time fixed and dynamic priority driven scheduling algorithms: theory and experience. RR-3081, INRIA, France.

[Jacquet et al. 1994]
Jacquet, P. Muhlethaler, P. Rivierre, N. Sept. 1994. Collision Detection in HIPERLAN. IEEE. PIMRC/WCN, The Hague. 875-879.

[Jacquet et al. 1996]
- Jacquet, P. Minet, P. Muhlethaler, P. Rivierre, N. 1996. Data Transfer for HIPERLAN. Wireless Personal Communication Journal. Kluwer Acaademic Publisher. 4: 65-80.
- Jacquet, P. Minet, P. Muhlethaler, P. Rivierre, N. 1996. Priority and collision detection with active signaling. Wireless Personal Communications Journal. Kluwer Academic Publisher. 4: 11-25.
- Jacquet, P. Minet, P. Muhlethaler, P. Rivierre, N. 1996. Increasing reliability in cable-free radio LANs low level forwarding in HIPERLAN. Wireless Personal Communications Journal. Kluwer Acaademic Publisher. 4: 51-63.

Bibliographie générale

[Agrawala et al. 1994]
Agrawala, A. K. Saksena, M. C. Yuan, X. 1994. A decomposition approach to Non-Preemptive Real-Time Scheduling. Real-Time Systems, 6, 7-35.

[Audsley 1991]
Audsley, N. C. 1991. Optimal priority assignment and Feasibility of static priority tasks with arbitrary start times. YCS 164, Dept. Comp. Science Report, University of York.

[Audsley 1995]
Audsley, N. C. Burns A., Davis, R. I. Tindell, K. Welling, A. J. 1995. Fixed Priority Pre-emptive Scheduling: An Historical Perspective. Real-Time Systems, 8, 173-198.

[Baker 1991]
Baker, T. P. 1991. Stack-Based Scheduling of Real-Time Processes. Real-Time Systems, 3, 67-99.

[Baruah et al. 1990]
Baruah, S. Howell, R. R. Rosier, L. Dec. 1990. Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic Real-Time tasks on one processor. Real-Time Systems, 2, 301-324.

[Baruah et al. 1990b]
Baruah, S. Mok, A. Rosier, L. Dec. 1990b. Preemptively Scheduling Hard-Real-Time Sporadic Tasks on One Processor. 11th Real-Time Systems Symposium. 182-190.

[Burns et al. 1994]
Burns, A. Tindell, K. Wellings, A. J. 1994. An extendible approach for analyzing fixed priority hard real time tasks. Real-Time Systems, 6, 133-151.

[Burns et al. 1995]
Burns, A. Tindell, K. Wellings, A. J. 1995 Analysis of hard real-time communications. Real-Time Systems, 9, 147-171.

[Cardeira 1994]
Cardeira, C. 1994. Ordonnancement temps réel par réseaux de neurones. Thèse de Doctorat, Institut National Polytechnique de Lorraine.

[Chen and Lin1990]
Chen, M. Lin, K. 1990. Dynamic Priority Ceilings: A Concurrency Control Protocol for Real-Time Systems. Real-Time Systems, 2, 325-345.

[Clark and Tindell 1994]
Clark, J. Tindell, K. Apr. 1994. Holistic Schedulability Analysis for Distributed Hard Real-Time Sustems. Microprocessing and Microprogramming Vol. 40, No. 2-3. 117-134.

[Dertouzos 1974]
Dertouzos, M. 1974. Control Robotics: the procedural control of physical processors. Proceedings of the IFIP congress, 807-813.

[Garey and Johnson 1979]
Garey, M. R. Johnson, D. S. 1979. Computer and Intractability, a Guide to the Theory of NP-Completeness. Ed. W. H. Freeman and Company, San Francisco.

[George et al. 1995]
George, L. Muhlethaler, P. Rivierre, N. 1995. Optimality and Non-Preemptive Real-Time Scheduling Revisited. RR-2516, INRIA, France.

[George et al. 1996]
George, L. Rivierre, N. Spuri, M. 1996. Preemptive and non-preemptive real-time uniprocessor scheduling. RR-2966, INRIA, France.

[George 1998]
George, L.. 1998. Ordonnancement en-ligne temps réel critique dans les systèmes distribués. Thése de Doctorat, Université de Versaille St Quentin en Yvelines.

[Hansson et al. 1994]
Hansson, H. Tindell, K. Wellings, A. J. 1994. analyzing real-time communications: controller Area Networks (CAN). Real-Time Systems. 259-263.

[Hermant et al. 1995]
Hermant, J.F. Le Lann, G. Rivierre, Oct. 1995. A general approach to real-time scheduling over broadcast channels. IEEE/INRIA, ETFA'95. 191-204.

[Hermant et al. 1996]
Hermant, J.F. Leboucher, L. Rivierre, N. 1996. Real-time fixed and dynamic priority driven scheduling algorithms: theory and experience. RR-3081, INRIA, France.

[Hermant 1998]
Hermant, J.F. 1998. Analyse d'algorithmes d'ordonnancement temps réel pour systèmes répartis. Thèse de Doctorat, Université Pierre et Marie Curie, Paris VI.

[Howell et al. 1995]
Howell, R. R. Venkatrao, M. K. Feb. 1995. On non-preemptive scheduling of recurring tasks using inserted idle time. Information and computation Journal, Vol. 117, Number 1.

[Jacquet et al. 1994]
Jacquet, P. Muhlethaler, P. Rivierre, N. Sept. 1994. Collision Detection in HIPERLAN. IEEE. PIMRC/WCN, The Hague. 875-879.

[Jacquet et al. 1996]
Jacquet, P. Minet, P. Muhlethaler, P. Rivierre, N. 1996. Data Transfer for HIPERLAN. Wireless Personal Communication. Kluwer Acaademic Publisher. 4: 65-80.

[Jeffay et al. 1991]
Jeffay, K. Stanat, D. F. Martel, C. U. Dec. 1991. On Non-Preemptive Scheduling of Periodic and Sporadic Tasks. IEEE Real-Time Systems Symposium, San-Antonio. 129-139.

[Jeffay et al. 1994]
Jeffay, K. Smith, F. D. Stone, D. L. 1994. Transport and display mechanisms for multimedia conferencing across packet switched networks. Comp. Networks, 26:1281-1304.

[Joseph and Pandya 1986]
Joseph, M. Pandya, P. 1986. Finding response times in a real-time system. BCS Com. Jour. 5(29) 390-395

[Katcher et al. 1993]
Katcher, D. I. Lehoczky, J. P. Strosnider, J. K. Apr. 1993, Scheduling models of dynamic priority schedulers. Research Report CMUCDS-93-4, Carnegie Mellon University, Pittsburgh.

[Kim and Naghibdadeh 1980]
Kim, K. H. Naghibzadeh, M. 1980. Prevention of task overruns in real-time non-preemptive multiprogramming systems. Proc. of Perf., Assoc. Comp. Mach. 267-276,

[Le Lann 1996]
Le Lann, G. 1996. A methodology for designing and dimensioning critical complex computing systems. IEEE Intl. symposium on the engineering of computer based systems (Germany). 3320-339.

[Le Lann 1996b]
Le Lann, G. 1996b. The Ariane 5 flight 501 Failure - A System engineering pespective. 10th IEEE Intl. ECBS Conf. (Monterey). 8 p.

[Leboucher and Stefani 1995]
Leboucher, L. Stefani, J. B. Nov 1995 Admission control for end-to-end distributed bindings. COST231, Lectures Notes in Computer Science, Vol 1052, 192-208.

[Leboucher 1997]
Leboucher, L. 1997. Fondement théorique de l'ordonnancement en ligne. Note technique CNET DTL/ASR/5238.

[Leboucher 1998]
Leboucher, L. 1998. Algorithmique et modélisation logique des systèmes temps réel. Thèse de doctorat de l'ENST.

[Lehoczky 1990]
Lehoczky, J. P. Dec. 1990. Fixed priority scheduling of periodic task sets with arbitrary deadlines. Proceedings 11th IEEE Real-Time Systems Symposium, 201-209.

[Lehoczky et al. 1990b]
Lehoczky, J.P. Rajkumar, Sha, L. R. 1990. Priority Inheritance Protocols: An Approach to Real-Time Synchronization," IEEE Transactions on Computers, 39(9): 1175-1185.

[Leung and Merril 1980]
Leung, J. Y. T. Merril, M. L. 1980. A note on preemptive scheduling of periodic, Real Time Tasks. Information processing Letters. 11(3): 115-118.

[Leung and Whitehead 1982]
Leung, J. Y. T. Whitehead, J. 1982. On the complexity of fixed-priority scheduling of periodic, real-time tasks. Performance Evaluation (Netherlands), 2(4): 237-250.

[Liu and Layland 1973]
Liu, C. L Layland, J. W. Jan. 1973. Scheduling Algorithms for multiprogramming in a Hard Real Time Environment. Journal of the Association for Computing Machinery, 20(1): 40-61.

[Lynch 1996]
Lynch, N. A. 1996. Distributed Algorithms. Morgan Kaufman Pub. ISBN 1-55860-348-4, 872p.

[Ma 1984]
Ma, P. R. Jan. 1984. A model to solve Timing-Critical Application Problems in Distributed Computing Systems. IEEE Computer, Vol. 17: 62-68.

[Mok 1983]
Mok, A. 1983. *Fundamental Design Problems for the Hard Real-Time Environments*. MIT/LCS/TR297 PhD.

[ODP 1995]
ITU-T. ISO/IEC. Recommendation X.903. International Standard 10746-1, 2, 3, 4. 1995. ODP Reference Model Architecture.

[OMG 1995]
The Common Object Request Broker Architecture and Specification. Revision 2.0. Object Management Group. Inc. Framingham. MA. USA. July 1995.

[Ramamritham and Stankovic 1987]
Ramamritham, K. Stankovic, J. A. Zhao, W. May 1987. Scheduling Task with Resource requirements in a Hard Real-Time System. IEEE Trans. on Soft. Eng., Vol. SE-13, No. 5, 564-577.

[Ripoll et al. 1996]
Ripoll, I. Crespo, A. Mok, A.K. 1996. Improvement in Feasibility Testing for Real-Time Tasks. Real-Time Systems, 11, 19-39.

[Roads Vehicles 1992]
Road Vehicles 1992. Interchange of Digital information - Controller Area Network (CAN) - for high speed communication, ISO DIS 11898.

[Shin and Zheng 1994]
Shin, K. G. Zheng, Q. 1994. On the Ability of Establishing Real-Time Channels in Point-to-Point Packet-Switched Networks. IEEE Transactions on Communications, 42(2/3/4).

[Spuri 1995]
Spuri, M. 1995. Earliest Deadline scheduling in real-time systems. Doctorate dissertation, Scuola Superiore S. Anna, Pisa,

[Spuri 1996]
Spuri, M. 1996. 1996. Analysis of deadline scheduled real-time systems. RR-2772, INRIA, France.

[Spuri 1996b]
Spuri, M. 1996b. 1996. Holistic analysis for deadline scheduled real-time distributed systems. RR-2873, INRIA, France.

[Tindell 1995]
Tindell, K. Feb. 1995. Holistic Schedulability Analysis for Distributed Hard Real-Time Systems. Euromicro Journal, Special Issue on Embedded Real-Time Systems,

[Yuan 1991]
Yuan, X. 1991. A decomposition approach to Non-Preemptive Scheduling on a single ressource. Ph.D. thesis, University of Maryland, College Park, MD 20742.

[Zheng 1993]
Zheng, Q. 1993. Real time fault tolerant communication in computer network. Univ. of Michigan, Ph. D. Dissertation.

Ordonnancement temps réel centralisé, les cas préemptifs et non-préemptifs

Résumé : La théorie de l'ordonnancement temps réel offre peu de critères de choix parmi les algorithmes pouvant résoudre un même problème. De nombreux résultats existent bien sûr, mais qui cherchent plus à prendre en compte des contraintes spécifiques qu'à se doter de critères de comparaison. A l'inverse, nous nous en tenons ici au "cas d'école" centralisé, préemptif/non-préemptif, avec des trafics à densités d'activation maximales de tâches. Sur ce modèle générique, nous proposons une synthèse des calculs de pires temps de réponse de tâches, de faisabilité des trafics et d'optimalité des algorithmes, ainsi que quelques éléments de comparaison (dominance, efficacité) des algorithmes.

La synthèse proposée nous conduit à utiliser certaines symétries sur les cas préemptifs/non-préemptifs, priorités fixes/dynamiques pour identifier des résultats manquants et/ou améliorations possibles. Avec l'algorithme à priorités dynamiques *EDF*, nous introduisons ainsi les *deadline busy periods* pour préciser les intervalles d'étude ou les calculs de pires temps de réponse. Nous précisons de même certaines restrictions sur l'optimalité des algorithmes non-préemptifs et/ou à priorités fixes.

L'optimalité s'avérant une propriété délicate, nous utilisons un critère d'efficacité pour comparer l'intérêt des algorithmes. *EDF* est le plus efficace, car optimal dans tous les cas. Des minorations existent sur l'efficacité de tout algorithme à priorités fixes que nous spécialisons pour *DM*, un algorithme à priorités fixes dominant. Les résultats obtenus confirment et précisent les conditions dans lesquelles il devient pertinent d'utiliser *DM* à la place de *EDF :* lorsque les trafics sont homogènes, lorsque les périodes des tâches sont petites face à leurs échéances ou, dans une moindre mesure, lorsque ces mêmes périodes deviennent grandes face aux échéances. Des applications numériques illustrent ces résultats.

Dans le cas non-préemptif non-oisif, les calculs de faisabilité et de pires temps de réponse s'adaptent du cas préemptif en tenant compte des inversions de priorités. Les résultats d'optimalité s'avèrent par contre limités et l'intérêt d'*EDF* en terme d'efficacité à relativiser. Les coûts des conditions de faisabilité sont enfin comparables à ceux connus en préemptif.

Mots clés: ordonnancement, temps réel, *EDF*, *DM*, faisabilité, optimalité, efficacité, préemptif, non-préemptif.

Centralized real-time scheduling, preemptive and non-preemptive cases

Abstract : Scheduling theory, regarding hard-real-time, has been widely studied in the last twenty-five years. Within the plethora of results, few of them are oriented towards a performance comparison of the algorithms. Rather than taking account new specific constraints, we focus on a generic task model for the centralized, preemptive/non-preemptive, fixed/dynamic priority cases.

In this context, we first propose to synthesis the traditional real-time approach for feasibility and optimality analysis. This allows us to pinpoint some possible generalizations (e.g. the worst-case response time with *Earliest Deadline First* in the non-preemptive case) and some restrictions on the optimality property (e.g. for the non-preemptive and/or fixed priority algorithms).

In order to overcome these restrictions on optimality, we make use of an efficiency criteria that measures the proximity to optimality with algebraical structures. *Earliest Deadline First* being the most efficient algorithm (optimal in any case), we extend preliminary results to precise when it becomes relevant to use *Deadline Monotonic* (one of the best fixed priority algorithm) instead of *Earliest Deadline First*. Some numerical applications are given to illustrate these results.

We also pinpoint that the theoretical advantage of *Earliest Deadline First* in the preemptive case (e.g. in presence of heterogeneous traffics) is not so significant in the non-preemptive case.

Key-words : scheduling, real-time, *EDF*, *DM*, feasibility, optimality, efficiency, preemptive, non-preemptive.